EXHIBIT 10

Europäisches Patentamt
European Patent Office
Office européen des brevets

Closure of the procedure in respect of application No. 04021916.4 - 2211 19.12.09

1. The procedure in respect of the above application is closed for the following reason:

   ADWI 11/09.09.09 The time limit under Rule 112(2) EPC has expired.
   No request for a decision under Rule 112(2), or for further processing under Article 121 EPC or for re-establishment of rights under Article 122 EPC has been filed.

2. The EPASYS situation has been verified in respect of item 1:

   DFIL: 15.09.04
   NOAP: ////
   RDEC: ////
   RFPR: //
   REES: ///

   XX
   REFU 3/ADWI 3 and DEAD 1 coded. Date of legal effect 6.8.2009.

3. Position regarding **fees**:

FFEE01	001	00018124	15.09.04	EUR	125,00
FEFS01	002	00018124	15.09.04	EUR	690,00
DEST03	005	00310618	08.04.06	EUR	240,00
EXAM02	006	00310618	08.04.06	EUR	1 490,00
CLMS(2)	015	00018124	15.09.04	EUR	280,00
RFEE 03	033	00592324	11.09.06	EUR	400,00
RFEE 04	034	00601368	12.09.07	EUR	425,00
RFEE 05	035	00758871	26.09.08	EUR	700,00
ASOC03	055	00018124	15.09.04	EUR	20,00

   ☐ 3.1 Form **2058A** submitted to 1st examiner (if applicable)

   ☐ 3.2 Refund(s) ordered:

   ☐ 75% Exam fee ☐ Other fees: _____

4. Mark "DEAD" on the paper file and:

   ☑ Check whether a divisional application is pending and if so attach the DEAD file to it.
   ☑ Any **models** still in the Office's possession were returned on _____
   (for dealing with models, please refer to Fil d'Ariane).
   ☐ Submit it to 1st examiner if a request for feedback is present.
   ☑ Keep paper file in file store (separate place) until next action for file destruction.

21-12-2009 Lambert, Karine

Date Formalities Officer

To 1st examiner/Director for information : Milasinovic G room 7671

Europäisches Patentamt
European Patent Office
Office européen des brevets

European Patent Office
80298 MUNICH
GERMANY
Tel. +49 (0)89 2399 - 0
Fax +49 (0)89 2399 - 4465

For any questions about this communication:
Tel.:+31 (0)70 340 45 00

Banzer, Hans-Jörg
Kraus & Weisert
Patent- und Rechtsanwälte
Thomas-Wimmer-Ring 15
80539 München
ALLEMAGNE

Date: 04.11.09

Reference	Application No./Patent No.
14738EP /nh	04021916.4 - 2211 / 1515229

Applicant/Proprietor: Trigence Corp.

Notice drawing attention to Rule 51(2) EPC, Article 2 No. 5 of the Rules relating to Fees,
- Payment of the renewal fee plus additional fee -

The renewal fee for the 06. year fell due on 30.09.09 unless this date falls within the period covered by an interruption of the proceedings in accordance with Rule 142(1) EPC, or a request for re-establishment of rights is pending (Art. 122, R. 51(4) EPC).

The current rate of the renewal fee amounts to **EUR 900,00** (see current Schedule of fees and costs).

The renewal fee was not paid by the due date.
The renewal fee may still be validly paid **up to the last day of the sixth calendar month** following the due date, provided that the additional fee (**50%** of the renewal fee) is paid at the same time, see OJ EPO 2008, 5.

Within the above period, which cannot be extended, the following fees are to be paid:

Renewal fee for the 06. year:	EUR	900,00
Additional fee:	EUR	450,00
TOTAL AMOUNT	EUR	1.350,00

If the renewal fee and the additional fee are not paid in due time, the European patent application shall be deemed to be withdrawn (Art. 86(1) EPC).

Note to users of the automatic debiting procedure
The normal time limit for payment of the above renewal fee had already expired when the automatic debit order was received. The renewal fee and the surcharge will be debited automatically on the last day of the six-month period (Supplement to OJ EPO 3/2009).

For the Examining Division

Europäisches Patentamt • European Patent Office • Office européen des brevets

Note
The Schedule of fees and costs of the EPO is published periodically as a Supplement to the Official Journal of the EPO. The current version is also available on the EPO website, together with a link to the epoline® facilities for viewing and downloading fees and searching for individual fees, both current and previous.

Europäisches Patentamt
European Patent Office
Office européen des brevets

European Patent Office
80298 MUNICH
GERMANY
Tel. +49 (0)89 2399 - 0
Fax +49 (0)89 2399 - 4465

Banzer, Hans-Jörg
Kraus & Weisert
Patent- und Rechtsanwälte
Thomas-Wimmer-Ring 15
80539 München
ALLEMAGNE

Formalities Officer
Name: Lambert
Tel.: 7574
or call:
+31 (0)70 340 45 00

De Poerck, Stéphanie

Date: 09-09-2009

Reference	Application No./Patent No.
14738EP /nh	04021916.4 - 2211 / 1515229
Applicant/Proprietor: Trigence Corp.	

Noting of loss of rights pursuant to Rule 112(1) EPC

The European patent application is deemed to be withdrawn under Article 94(4) EPC, because the invitation to file observations on the communication from the Examining Division dated 26.01.09 was not complied with.

Means of redress

Request for a decision (R. 112(2) EPC)
If the applicant considers that the finding of the European Patent Office is inaccurate, he may, within a (non-extendable) period of **two months** after notification of this communication, apply in writing for a decision on the matter. The application can only lead to the finding being reversed if this does not actually correspond to the factual or legal situation.

Further processing (Art. 121 EPC)
The legal consequence of the failure to observe the time limit shall be deemed not to have ensued if, within a (non-extendable) period of **two months** after notification of this communication, further processing is requested by payment of the fee prescribed under Article 2(12) of the Rules relating to Fees and the omitted act is completed (R. 135(1) EPC).

Important note to users of the automatic debiting procedure
The fee for further processing will be debited automatically on the day on which the above-mentioned omitted act is completed (see Arrangements for the automatic debiting procedure, Supplement to OJ EPO 3/2009).

For the Examining Division

Europäisches Patentamt • European Patent Office • Office européen des brevets

Registered letter
EPO Form 2021A 12.07 04.09.09 ADWI 11

Europäisches Patentamt
European Patent Office
Office européen des brevets

European Patent Office
80298 MUNICH
GERMANY
Tel. +49 (0)89 2399 - 0
Fax +49 (0)89 2399 - 4465

Banzer, Hans-Jörg
Kraus & Weisert
Patent- und Rechtsanwälte
Thomas-Wimmer-Ring 15
80539 München
ALLEMAGNE

For any questions about this communication:
Tel.:+31 (0)70 340 45 00

Date
25.05.09

Reference	Application No./Patent No.
14738EP /nh	04021916.4 - 2211 / 1515229

Applicant/Proprietor
Trigence Corp.

Extension of time limit pursuant to Rule 132(2) EPC

Examination procedure

With reference to your request, the time limit for replying to the communication according to Article 94(3) EPC dated 26.01.09 has been extended

by 2 months

to a total of 6 months

from the date of notification of the above-mentioned communication.

Please note: To the extent that your request exceeded the above extension, your request has been refused.

Note
The granting of extensions to time limits is governed by the Implementing Regulations to the EPC and the Guidelines for Examination in the EPO, part E-VIII, 1.6.

If no reply to the communication is received in due time, the European patent application will be deemed to be withdrawn (Art. 94(4) EPC).

For the Examining Division

Europäisches Patentamt • European Patent Office • Office européen des brevets

Kraus & Weisert
Patent- und Rechtsanwälte
European Patent and Trademark Attorneys

EPO - Munich
62
14. Mai 2009

Kraus & Weisert · Thomas-Wimmer-Ring 15 · 80539 München

European Patent Office
Erhardtstraße 27
80469 Munich

Dr. Walter Kraus (– 2002)
Dr.-Ing. Annekäte Weisert (– 2002)
Dr. Thomas Albrecht
Dipl.-Ing. Hans-Jörg Banzer
Dr. Holger Adam
Dr. Inge Hiebl
Dr. Claus Beckmann
Dr. Florian Bertsch
Dr. Andreas Sticht
Dr. Markku Schwarz
Dr. Florian Meier
Dr. Ferdinand Nielsen
Rechtsanwalt

Unser Zeichen / Our Ref.: 14738EP /sm

Bitte in der Antwort angeben
Please refer to in your reply

May 14, 2009

Re: European Patent Application 04 021 916.4-2211
Trigence Corp.

It is politely requested to extend the term for answering the Communication pursuant to Article 94(3) EPC dated January 26, 2009 by two months, i.e. until

July 26, 2009.

Should the requested extension of term not be granted, a corresponding notification (possibly by phone) is politely requested.

Dipl.-Ing. Hans-Jörg Banzer
European Patent Attorney

Thomas-Wimmer-Ring 15
80539 München

Tel. +49 (0)89 / 290 60-0
Fax +49 (0)89 / 290 60-111
office@kraus-weisert.de
www.kraus-weisert.de

HypoVereinsbank München
Kto.-Nr.: 667 340 080
BLZ: 700 202 70
IBAN: DE96 7002 0270 0667 3400 80
SWIFT: HYVEDEMM

Postbank München
Kto.-Nr.: 851 02-809
BLZ: 700 100 80
IBAN: DE67 7001 0080 0085 1028 09
SWIFT: PBNKDEFF

The examination is being carried out on the **following application documents**:

Description, Pages

2, 2a	received on	18.08.2006	with letter of	18.08.2006

Claims, Numbers

1-17	filed in electronic form on	02.10.2008

Drawings, Sheets

1/17-13/17, 15/17-17/17	as originally filed			
14/17	received on	09.11.2004	with letter of	09.11.2004

1 Article 123(2) EPC

1.1 The term *"wherein **application software has the identity** in the form of the IP address, host name or MAC address of the container that it is associated with"* is not disclosed in the description and is therefore claim 1 is **not allowable** with regard to Article 123(2) EPC. The IP address, MAC address and host name which are associated to a container are merely **a part of the unique ID** of an application, since a container can host multiple applications in which case each of those applications would have the same identifier **(see the description, page 7, line 16-17**, "...a group of applications within a single container share a same IP address, unique to that container..." and **page 13, line 16-18**, "The type of information required in order to provide an application associated with a container a unique identity **includes** items such as an IP address, MAC address and host name"**)**.

2 Clarity

2.1 The vague and imprecise statement in the description on page 10, line 3-4 implies that the subject-matter for which protection is sought may be different to that defined by the claims, thereby resulting in a **lack of clarity** of the claims (Article 84 EPC) when the description is used to interpret the claims (see Guidelines C-III, 4.4). This statement should therefore be amended to remove this inconsistency.

3 Remarks to the letter of reply, dated 2.10.2008

3.1 The unique identifer, **as specified** by the independent claims 1 and 15, is merely used a **general means** of identification and thus represents an **equal alternative** to the common practise of providing unique identifiers for entities in a computing system.

3.2 The unique identity of the container and the applications, it appears, is directed to a unique network identifier in the form of a MAC address, IP address or host name and **not to a general unique identifier**. As disclosed in the description it appears that the unique ID of the container is used to allow *"other applications to locate a containerized service in a consistent manner independent of which computer platform the containerized service is located on"* **(page 13, line 11-13)**. This feature, however, is **not part of the independent claims and thus cannot contribute to the inventive step**.

3.3 The applicant claims that **D1** doesn't disclose *"a unique root file system"* for each container. This, however, is not true since **D1** does disclose that *"The present invention provides a **virtual file system** much like the virtual registry described above. Before the application starts, the present invention can load a list of file system changes, including files to hide and files to add to the virtual environment or files to redirect to another within the virtual environment"* **(paragraph 40)**.

4 Conclusion

4.1 The applicant is invited to file new claims which take account of the above comments.

4.2 When filing amended claims the applicant should at the same time bring the description into conformity with the amended claims. Care should be taken during revision, especially of the introductory portion and any statements of problem or advantage, not to add subject-matter which extends beyond the content of the application as originally filed (Article 123(2) EPC).

4.3 In order to facilitate the examination of the conformity of the amended application with the requirements of Article 123(2) EPC, the applicant is requested to clearly

identify the amendments carried out, irrespective of whether they concern amendments by addition, replacement or deletion, and to indicate the passages of the application as filed on which these amendments are based.

4.4 The applicant is requested to effect the amendments by filing replacement pages for only those pages which have been amended. Unnecessary recasting of the description should be avoided. An amended abstract is not required.

4.5 Moreover, it is considered as appropriate in the present case to draft the new independent claim in the two-part form as required by Rule 43(1) EPC, whereby the features known in combination from document **D1** should be placed in the preamble. If the applicant is of the opinion that a two-part form of the claim would be inappropriate he is invited to provide reasons in his reply. In addition, the applicant should ensure that it is clear from the description which features of the subject-matter of the new independent claim are known from document **D1**; see Guidelines C-III,2.3.2.

Europäisches Patentamt
European Patent Office
Office européen des brevets

European Patent Office
80298 MUNICH
GERMANY
Tel: +49 89 2399 0
Fax: +49 89 2399 4465

Banzer, Hans-Jörg
Kraus & Weisert
Patent- und Rechtsanwälte
Thomas-Wimmer-Ring 15
80539 München
ALLEMAGNE

Formalities Officer
Name: Lambert, Karine
Tel: +49 89 2399 - 7574
or call
+31 (0)70 340 45 00

Substantive Examiner
Name: Milasinovic, Goran
Tel: +49 89 2399 - 5611

Application No.	Ref.	Date
04 021 916.4 - 2211	14738EP /nh	26.01.2009
Applicant: Trigence Corp.		

Communication pursuant to Article 94(3) EPC

The examination of the above-identified application has revealed that it does not meet the requirements of the European Patent Convention for the reasons enclosed herewith. If the deficiencies indicated are not rectified the application may be refused pursuant to Article 97(2) EPC.

You are invited to file your observations and insofar as the deficiencies are such as to be rectifiable, to correct the indicated deficiencies within a period

of 4 months

from the notification of this communication, this period being computed in accordance with Rules 126(2) and 131(2) and (4) EPC. One set of amendments to the description, claims and drawings is to be filed within the said period on separate sheets (R. 50(1) EPC).

Failure to comply with this invitation in due time will result in the application being deemed to be withdrawn (Art. 94(4) EPC).

Europäisches Patentamt • European Patent Office • Office européen des brevets

Milasinovic, Goran
Primary Examiner
For the Examining Division

Enclosure(s): 3 page/s reasons (Form 2906)

Claims:

1. In a system having a plurality of servers (10a, 10b) with operating systems that differ, operating in disparate computing environments, wherein each server includes a processor (13) and an operating system including a kernel (12), a set of associated local system files compatible with the processor (13), a method of providing at least some of the servers in the system with secure, executable, applications (21a-21f) related to a service, wherein the executable applications (21a-21f) may be executed in a secure environment, wherein the executable applications (21a-21f) each include an object executable by at least some of the different operating systems for performing a task related to the service, the method comprising the steps of:

storing in memory accessible to at least some of the servers a plurality of secure containers (20a-20c) of application software (21a-21f), each container (20a-20c) of application software having its own unique identity in the form of an IP address, host name or MAC address and comprising one or more of the executable applications (21a-21f) and a set of associated system files required to execute the one or more executable applications (21a-21f), for use with a local kernel (12) residing permanently on one of the servers; wherein the set of associated system files are compatible with a local kernel (12) of at least some of the plurality of different operating systems, the containers (20a-20c) of application software excluding a kernel, wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files that remain resident on the server wherein said associated system files utilized in place of the associate local system files are copies or modified copies of the associated local system files that remain resident on the server, and wherein files cannot be shared between the plurality of secure containers of application software, and wherein each of the containers has a unique root file system that is different from the operating system's root file system, and wherein application software (21a-21f) has the identity in the form of the IP address, host name or MAC address of the container that it is associated with so that executable applications can execute in

conjunction with an operating system that said executable application is not originally intended to operate with and would otherwise not function properly.

2. A method as defined in claim 1, wherein each container (20a-20c) has an execution file associated therewith for starting the one or more applications (21a-21f), and wherein the execution file includes instructions related to an order in which executable applications within will be executed, and wherein in operation when applications are executed, applications within a container (20a-20c) have no access to system files or applications in other containers (20a-20c) or system files within the operating system during execution thereof if those applications or system files are read/write files.

3. A method as defined in claim 1 or claim 2 further comprising the step of pre-identifying applications and system files required for association with the one or more containers (20a-20c) prior to said storing step, and modifying at least some of the system files to provide an association with a container specific identity assigned to the container.

4. A method as defined in any one of claims 1, 2, or 3 comprising the step of assigning said unique associated identity to each of a plurality of the containers (20a-20c).

5. A method as defined in any one of claims 1 through 4, wherein the one or more applications (21a-21f) and associated system files are retrieved from a computer system having a plurality of secure containers (20a-20c).

6. A method as defined in any one of claims 2 through 5 wherein server information related to hardware resource usage including at least one of CPU memory, network bandwidth and disk allocation is associated with at least some of the containers (20a-20c) prior to the applications (21a-21f) within the containers (20a-20c) being executed.

7. A method as defined in any of claims 2 through 6, wherein containers include files stored in network file storage (82), and parameters forming descriptors of containers stored in a separate location (84).

8. A method as defined in any of claims 1 through 7 comprising the step of creating containers (20a-20c) prior to said step of storing containers (20a-20c) in memory, wherein the step of creating containers comprises the steps of:

a) running an instance of a service on a server,

determining which files are being used, and,

copying applications and associated system files to memory;

or,

(b) using a skeleton set of system files as a container starting point and installing applications into that set of files.

9. A method as defined in any of claims 1 to 8 further comprising the step of installing a service on a target server selected from one of the plurality of servers (10a, 10b), wherein the step of installing the service includes the steps of:

using a graphical user interface, associating a unique icon representing a service with an unique icon representing a server for hosting applications related to the service and for executing the service, so as to cause the applications to be distributed to, and installed on the target server.

10. A method as defined in claim 9 wherein the target server and the graphical user interface are at remote locations, or wherein the graphical user interface is installed on a computing platform, and wherein the computing platform is a different computing platform than the target server.

11. A method as defined in any of claims 9 or 10, wherein the step of associating includes the step of relatively moving the unique icon representing the service to the unique icon representing a server.

12. A method as defined in any of claims 9 through 11 further comprising the step of: de-installing a service from a server, comprising the steps of:

displaying the unique icon representing the service;

displaying the unique server icon representing the server on which the service is installed; and utilizing the icon representing the service and the icon representing the server to initiating the de-installation of the selected software application from the selected server.

13. A method according to any of claims 9 through 12 further comprising the step of separating the icon representing the service from the icon representing the server.

14. A method according to claim 13 further comprising the step of copying data file changes specific to the service back to a storage medium from which the data file changes originated prior to installation.

15. A computing system for performing a plurality of tasks each comprising a plurality of processes comprising:

a plurality of secure stored containers (20a-20c) of associated files accessible to, and for execution on, one or more servers (10a, 10b), each container (20a-20c) being mutually exclusive of the other, such that read/write files within a container (20a-20c) cannot be shared with other containers, each container (20a-20c) of files having its own unique identity associated therewith, said identity comprising at least one of an Internet Protocol address, a host name, and a MAC address; wherein, the plurality of files within each of the plurality of containers comprise one or more application programs (21a-21f) including one or more processes, and associated system files for use in executing the one or more processes, each container (20a-20c) having its own execution file associated therewith for starting one or more applications, in operation, each container (20a-20c) utilizing a kernel (12) resident on the server (10a, 10b) and wherein each container (20a-20c) exclusively uses a kernel (12) in an underlying operation system in which it is running and is absent its own kernel; and,
a run time module for monitoring system calls from applications associated with one or more containers and for providing control of the one or more applications.

16. A computing system as defined in claim 15, further comprising a scheduler comprising values related to an allotted time in which processes within a container (20a-20c) may utilize

predetermined resources and, wherein the run time module includes an intercepting module associated with the plurality of containers (20a-20c) for intercepting system calls from any of the plurality of containers (20a-20c) and for providing values alternate to values the kernel would have assigned in response to the system calls, so that the containers can run independently of one another without contention, in a secure manner, the values corresponding to at least one of the Internet Protocol address, the host name and the MAC address.

17. A computing system as defined in claim 16, wherein the run time module performs:
monitoring resource usage of applications executing;
intercepting system calls to kernel mode, made by the at least one respective application within a container, from user mode to kernel mode;
comparing the monitored resource usage of the at least one respective application with the resource limits; and,
forwarding the system calls to a kernel on the basis of the comparison between the monitored resource usage and the resource limits.

Claims:

1. In a system having a plurality of servers (10a, 10b) with operating systems that differ, operating in disparate computing environments, wherein each server includes a processor (13) and an operating system including a kernel (12), a set of associated local system files compatible with the processor (13), a method of providing at least some of the servers in the system with secure, executable, applications (21a-21f) related to a service, wherein the executable applications (21a-21f) may be executed in a secure environment, wherein the executable applications (21a-21f) each include an object executable by at least some of the different operating systems for performing a task related to the service, the method comprising the steps of:

storing in memory accessible to at least some of the servers a plurality of secure containers (20a-20c) of application software (21a-21f), each container (20a-20c) of application software having its own unique identity in the form of an IP address, host name or MAC address and comprising one or more of the executable applications (21a-21f) and a set of associated system files required to execute the one or more executable applications (21a-21f), for use with a local kernel (12) residing permanently on one of the servers; wherein the set of associated system files are compatible with a local kernel (12) of at least some of the plurality of different operating systems, the containers (20a-20c) of application software excluding a kernel, ~~and~~ wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files that remain resident on the server ~~prior to said storing step~~wherein said associated system files utilized in place of the associate local system files are copies or modified copies of the associated local system files that remain resident on the server, and wherein files cannot be shared between the plurality of secure containers of application software, and wherein each of the containers has a unique root file system that is different from the operating system's root file system,~~-~~ and wherein application software (21a-21f) has the identity in the form of the IP address, host name or MAC address of the container that it is associated with so that executable

applications can execute in conjunction with an operating system that said executable application is not originally intended to operate with and would otherwise not function properly~~, and wherein application software (21a-21f) has an identity of a container that it is associated with~~.

2. A method as defined in claim 1, wherein each container (20a-20c) has an execution file associated therewith for starting the one or more applications (21a-21f), and wherein the execution file includes instructions related to an order in which executable applications within will be executed, and wherein in operation when applications are executed, applications within a container (20a-20c) have no access to system files or applications in other containers (20a-20c) or system files within the operating system during execution thereof if those applications or system files are read/write files.

3. A method as defined in claim 1 or claim 2 further comprising the step of pre-identifying applications and system files required for association with the one or more containers (20a-20c) prior to said storing step, and modifying at least some of the system files to provide an association with a container specific identity assigned to the container.

4. A method as defined in any one of claims 1, 2, or 3 comprising the step of assigning said unique associated identity to each of a plurality of the containers (20a-20c)~~, wherein the identity includes at least one of IP address, host name, and MAC address, a unique identifier of a network node~~.

5. A method as defined in any one of claims 1 through 4, wherein the one or more applications (21a-21f) and associated system files are retrieved from a computer system having a plurality of secure containers (20a-20c).

6. A method as defined in any one of claims 2 through 5 wherein server information related to hardware resource usage including at least one of CPU memory, network bandwidth and disk allocation is associated with at least some of the containers (20a-20c) prior to the applications (21a-21f) within the containers (20a-20c) being executed.

7. A method as defined in any of claims 2 through 6, wherein containers include files stored in network file storage (82), and parameters forming descriptors of containers stored in a separate location (84).

8. A method as defined in any of claims 1 through 7 comprising the step of creating containers (20a-20c) prior to said step of storing containers (20a-20c) in memory, wherein the step of creating containers comprises the steps of:

a) running an instance of a service on a server,

determining which files are being used, and,

copying applications and associated system files to memory;

or,

(b) using a skeleton set of system files as a container starting point and installing applications into that set of files.

9. A method as defined in any of claims 1 to 8 further comprising the step of installing a service on a target server selected from one of the plurality of servers (10a, 10b), wherein the step of installing the service includes the steps of:

using a graphical user interface, associating a unique icon representing a service with an unique icon representing a server for hosting applications related to the service and for executing the service, so as to cause the applications to be distributed to, and installed on the target server.

10. A method as defined in claim 9 wherein the target server and the graphical user interface are at remote locations, or wherein the graphical user interface is installed on a computing platform, and wherein the computing platform is a different computing platform than the target server.

11. A method as defined in any of claims 9 or 10, wherein the step of associating includes the step of relatively moving the unique icon representing the service to the unique icon representing a server.

12. A method as defined in any of claims 9 through 11 further comprising the step of: de-installing a service from a server, comprising the steps of:
displaying the unique icon representing the service;
displaying the unique server icon representing the server on which the service is installed;
and utilizing the icon representing the service and the icon representing the server to initiating the de-installation of the selected software application from the selected server.

13. A method according to any of claims 9 through 12 further comprising the step of separating the icon representing the service from the icon representing the server.

14. A method according to claim 13 further comprising the step of copying data file changes specific to the service back to a storage medium from which the data file changes originated prior to installation.

15. A computing system for performing a plurality of tasks each comprising a plurality of processes comprising:

a plurality of secure stored containers (20a-20c) of associated files accessible to, and for execution on, one or more servers (10a, 10b), each container (20a-20c) being mutually exclusive of the other, such that read/write files within a container (20a-20c) cannot be shared with other containers, each container (20a-20c) of files having its own unique identity associated therewith, said identity comprising at least one of an Internet Protocol address, a host name, and a MAC address~~, a unique identifier of a network node~~; wherein, the plurality of files within each of the plurality of containers comprise one or more application programs (21a-21f) including one or more processes, and associated system files for use in executing the one or more processes, each container (20a-20c) having its own execution file associated therewith for starting one or more applications, in operation, each container (20a-20c) utilizing a kernel (12) resident on the server (10a, 10b) and wherein each container (20a-20c) exclusively uses a kernel (12) in an underlying operation system in which it is running and is absent its own kernel; and,

a run time module for monitoring system calls from applications associated with one or more containers and for providing control of the one or more applications.

16. A computing system as defined in claim 15, further comprising a scheduler comprising values related to an allotted time in which processes within a container (20a-20c) may utilize predetermined resources and, wherein the run time module includes an intercepting module associated with the plurality of containers (20a-20c) for intercepting system calls from any of the plurality of containers (20a-20c) and for providing values alternate to values the kernel would have assigned in response to the system calls, so that the containers can run independently of one another without contention, in a secure manner, the values corresponding to at least one of the Internet Protocol address, the host name and the MAC address.

17. A computing system as defined in claim 16, wherein the run time module performs:
monitoring resource usage of applications executing;
intercepting system calls to kernel mode, made by the at least one respective application within a container, from user mode to kernel mode;
comparing the monitored resource usage of the at least one respective application with the resource limits; and,
forwarding the system calls to a kernel on the basis of the comparison between the monitored resource usage and the resource limits.

02.10.08

Kraus & Weisert

Patent- und Rechtsanwälte
European Patent and Trademark Attorneys

EPO - Munich
79
02. Okt. 2008

Kraus & Weisert · Thomas-Wimmer-Ring 15 · 80539 München

European Patent Office
Erhardtstraße 27
80469 Munich

Dr. Walter Kraus (– 2002)
Dr.-Ing. Annekäte Weisert (– 2002)
Dr. Thomas Albrecht
Dipl.-Ing. Hans-Jörg Banzer
Dr. Holger Adam
Dr. Inge Hiebl
Dr. Claus Beckmann
Dr. Florian Bertsch
Dr. Andreas Sticht
Dr. Horst Glaser
Dr. Markku Schwarz
Dr. Florian Meier
Dr. Ferdinand Nielsen
Rechtsanwalt

Unser Zeichen / Our Ref.: 14738EP HJB/AS/kr

Bitte in der Antwort angeben
Please refer to in your reply

October 2, 2008

Re: European Patent Application 04 021 916.4-2211
Trigence Corp.

Responsive to the Communication dated June 3, 2008.

I.

Enclosed new claims 1-17 are filed replacing the set of claims currently on file. To make it easier for the Examining Division to reconstruct the amendments carried out, a claim set where the changes compared to the currently pending claims are marked is additionally filed.

New claim 1 is based on present claim 1. It has been added that the identity is in the form of an IP address, a host name or a MAC address. This feature may for example be taken from claim 4 as filed.

Furthermore, the feature has been added that the associated system files utilized in place of the associated local system files are copies or modified copies of the

associated local system files that remain resident on the server. This feature can be taken from page 16, lines 19-20, page 16, lines 25-31, page 17, lines 16-18, page 18, lines 3-5, lines 18-20 and lines 24-25 of the application as filed which clearly show that the associated system files of the container are copied or modified copies of the associated local system files that remain resident on the server.

Furthermore, the feature has been added that files cannot be shared between the plurality of secure containers of application software. This amendment is based on page 12, lines 7-11 and in particular page 15, lines 10-13 and page 16, lines 19-20 of the application as filed.

Finally, the feature has been added that each of the containers has a unique root file system that is different from the operating system's root file system. This feature can be taken from page 7, lines 12-14 as well as page 19, lines 19-20 of the application as filed.

It should be noted that the above citations are only some examples where the amended features of new claim 1 are disclosed. Further explanations and support of the corresponding features generally may be found on page 11, line 26 – page 12, line 2, in particular the last two lines of this passage explaining the replacement of the system's file of the operating system by the corresponding system files of the container, page 12, lines 4-11 explaining the relationship between applications, the container and the operating system, page 12, lines 18-22, page 15, lines 8-13 further explaining the relationship between application and containers, page 16, lines 18-21, page 17, lines 14-18, and page 18, lines 10-21.

New claims 2 and 3 correspond to present claims 2 and 3. New claim 4 is based on present claim 4 with the feature of the IP address, host name and MAC address which is already defined in claim 1 having been cancelled.

New claims 5-17 correspond to present claims 5-17, wherein in claim 15 following the objection under item 2.1 of the above-referenced Official Communication the words "a unique identifier of a network node" has been cancelled. It should be noted that this feature was only introduced following an explicit proposal under

item 3.3 of the Official Communication dated May 5, 2006 where the Examining Division was of the opinion that the term "MAC address" is unclear. In applicant's view, "MAC address" is a fixed technical term very well known to persons skilled in the art and therefore need not be further defined.

In applicant's view, no modifications to the presently pending description are necessary to adapt the description to the wording of the new claims. However, in case the Examining Division deems such an adaptation necessary, it may be performed at short notice as soon as an agreement regarding the claims has been reached.

II.

In the above-referenced Official Communication dated June 3, 2008, it has already been acknowledged that document D1 does not disclose an identity comprising at least one of an IP address, a host name and a MAC address (see item 4.1). The corresponding feature now has also been introduced in new claim 1. Therefore, at least because of this feature also claim 1 is novel over prior art document D1.

III.

Furthermore, the subject-matter of new claims 1 and 15 is not only new, but also involves the necessary inventive step.

In this respect, the conclusions under item 3.1 of the Official Communication regarding implicit disclosures cannot be followed.

In particular, it is stated that it is implicit that each container of application software has its own unique identity and that allegedly it is also implicit that application software has an identity of a container that it is associated with. Even if it were accepted that it is necessary that each entity of a system can be identified in some manner, this in no way implies that each container has a separate identity and that the application software of a container has the identity of that container it

is associated with. The reasoning of the Examining Division that every operating system entity needs an identity would, even if accepted, only mean that the application software has some kind of identity, but would not imply that the application software has the identity of the container that it is associated with.

Moreover, the reasoning of the Examining Division under item 4 that it would be obvious to use the IP address, host name or MAC address as such an identifier cannot be followed. In particular, these elements as mentioned under item 4.4 of the Official Communication are usually used to identify elements in a network. There is no hint whatsoever in the prior art to use these elements as identifiers in a method or a computing system as defined in the present independent claims. In this respect, it should be noted that the reasoning in the Official Communication relies on the fact that every operating system entity needs an identity, but clearly an operating system is different from a network, and there is no incentive in the prior art whatsoever to use an IP address, a host name or a MAC address as identifier of an operating system, such that the two parts of the argumentation do not fit together. The differences between the containers of the present invention and the prior art are further enhanced by the additional features of new claim 1, for example that the root file systems of the containers are different from each other and from the root file system of the operating system. The above-mentioned differences will be explained below in some more detail:

The instant invention provides a system having plural, secure containers of application software that are capable of running under an operating system under which they were not intended to run. The containers of application software each have a unique network ID, such as an IP address, MAC address or host name; and software applications within each container assume the same unique ID (MAC address, IP address, or host name) of the container and this unique ID associated with a container and the applications that run within the container is different than the ID (IP address, MAC address; host name) of the operating system, under which the applications execute.

In D1, the operating system acquires a network ID such as an IP address, MAC address or host name and the application software uses the same network ID

used by the OS itself and passed on by the operating system to the application software.

Under item 3.1 the examiner states that containers having unique identities is implicit in the cited prior art reference, and regarding the feature that the application software has an identity of the container, the examiner also states that this is also implicit "since very operating system needs an identity".

In contradistinction, in this invention, (now defined in claim 1) the network ID, defined as the IP address, MAC address, or host name of the OS is different than that of the containers, and the network ID of each container is different than that of any other container, and software applications within a container use the network ID of that specific container in which they reside.

Another significant distinguishing aspect of the invention is that files residing in a container cannot be shared between the plurality of secure containers of application software, and each of the containers each has a unique root file system that is different from the operating system's root file system.

These features are not taught or suggested by D1 which does not teach a plurality of containers of application software having the features defined in amended claim 1 wherein each container has its own unique network ID and its own root file system that different from the operating system's root file system.

Claim 15 is said to lack inventiveness over the prior art in the Official Communication. Once again, the examiner has suggested that having containers of application software with unique network identifiers such that each container has a different identifier in the form of a MAC IP address, MAC address or host name is not explicitly stated in the prior art but is implicit. There is simply no suggestion of a system wherein each container of application software has its own unique network identifier. In contradistinction, in D1 the network identity of the OS is the network identity that is used by his software applications.

In summary, it has been shown that even by taking document D1 and his technical knowledge into account, a person skilled in the art could not arrive at the subject-matter of new claim 1 or new claim 15 without being inventive. Consequently, these claims also involve the required inventive step.

IV.

In summary, as has been shown above, the newly submitted claims are patentable over the prior art. A short appreciation of document D1 has already been incorporated with a prior response to an Official Communication, such that in applicant's view the present application is now ready for grant. In case, however, the Examining Division is still doubtful regarding the allowability of the present application, it is respectfully requested that a further Communication pursuant to Art. 94(3) EPC be issued. As an auxiliary measure, oral proceedings pursuant to Art. 116 EPC are requested should the Examining Division intend to refuse the present application.

Dr. Andreas Sticht
European Patent Attorney

Encls.:
New claims 1-17,
Marked-up version of the new claims

Europäisches Patentamt
European Patent Office
Office européen des brevets

04021916.4 - 2211 / 1515229 24.09.08

Client Database System (CDS) - clean up.

Application Nr.: 04021916.4

Following clean up action in CDS the entries concerning the **Representative for the applicant** have been amended and are now as follows:

Banzer, Hans-Jörg
Kraus & Weisert
Patent- und Rechtsanwälte
Thomas-Wimmer-Ring 15
80539 München
DE

Where appropriate, the Register of European Patents will be updated to show the amended details.

For questions please contact the Client Data Registration department of the European Patent Office in Munich, telephone +49 (0)89 2399 2780.

The examination is being carried out on the **following application documents**:

Claims, Numbers

1-17	received on	18.08.2006	with letter of	18.08.2006

Drawings, Sheets

1/17-13/17, 15/17-17/17	as originally filed			
14/17	received on	09.11.2004	with letter of	09.11.2004

1 Summary

1.1 The claims, filed with letter of 18.08.2006, do not meet the requirements of Article 123(2) EPC.

1.2 The present application does not meet the requirements of Article 52(1) EPC, because the subject-matter of claims 1-3 is **not new** and thus not allowable in the sense of Article 54(1) and (2) EPC.

2.3 Furthermore, the subject-matter of claims 4-17 is **not inventive** (Article 56).

2 Article 123(2) EPC

2.1 Claims 4 and 15 specify that an identity of a container includes *"a unique identifier of a network node"*. This feature is **not explicitly disclosed** in the description and thus claims 4 and 15 are **not allowable** with regard to Article 123(2) EPC.

3 Novelty of independent claim 1

3.1 Document **D1** is regarded as closest prior art. It discloses in the original wording of independent claim 1 (reference to the closest prior art is made in parentheses; the original wording of the claim is set in *italic font*):

In a system having a plurality of servers with operating systems that differ, operating in disparate computing environments, wherein each server includes a processor and an operating system including a kernel a set of associated local system files compatible with the processor **(implicit**, since this is a common IT structure**)**, *a method of providing at least some of the servers in the system with secure, executable, applications related to a service, wherein the executable applications may be executed in a secure environment* **(page 1, paragraph 4, line 3-4**, "...an operating system abstraction and protection layer..."**)**, *wherein the executable applications each include an object executable by at least some of the different operating systems for performing a task related to the service, the method comprising the steps of:*

storing in memory accessible to at least some of the servers a plurality of secure containers of application software, each container of application software **(page 1, paragraph 4, line 3-4**, "...an operating system abstraction and protection layer..."**)** *having it's own unique identity* **(implicit**, since every operating system entity needs an identity**)** *and comprising one or more of the executable applications and a set of associated system files required to execute the one or more applications, for use with a local kernel residing permanently on one of the servers* **(page 2, paragraph 23, line 7-9**, "...An application bundle is a group of processes..." and **page 2, paragraph 24, line 3-5**, "...serializing its configuration..."**)**;

wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems, the containers of application software excluding a kernel **(figure 1)**, *and*

wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files resident on the server prior to said storing step **(page 2, paragraph 13, line 7-13**, "...if an application attempts to change the version of a shared object like MSVCRT.DLL this change is localized to the application..."**)** *so that executable applications can execute in conjunction with an operating system that said executable application is not originally intended to operate with and would otherwise not function properly* **(page 2, paragraph 13, line 7-13**, "...if an application attempts to change the version of a shared object like MSVCRT.DLL this change is localized to the application...", see remarks in item 7 below**)**, *and wherein application software has an identity of a container that it is associated with* **(implicit**, since every operating system entity

needs an identity**)**.

3.2 Therefore, the subject-matter of claim 1 is **not new** and thus this claim is not allowable, Articles 52(1) and 54 EPC. Should the applicant be able to identify minor differences or amend the claim by such differences which overcome the above novelty objection, then still the claim can not be considered to be allowable for lack of inventive step, Articles 52(1) and 56 EPC.

4 Inventiveness of independent system claim 15

4.1 Document **D1** is regarded as closest prior art. It discloses in the original wording of independent system claim 15 (reference to the closest prior art is made in parentheses; the original wording of the claim is set in *italic font*; features not explicitly disclosed in the prior art are set ~~strikeout~~)

A computing system for performing a plurality of tasks each comprising a plurality of processes comprising:

a plurality of secure stored containers of associated files accessible to **(implicit**, since each application consists of one or more files**)**, *and for execution on, one or more servers, each container being mutually exclusive of the other, such that read/write files within a container cannot be shared with other containers* **(page 2, paragraph 19, line 1-4**, "...each application is able to run in a private context..." and **page 4 paragraph 40, line 3-4**, "...The present invention provides a virtual file system much like the virtual registry..."; like with the registry, where an application has it's own folders where it can store configuration keys and values, the virtual file system provides the applications own private file system**)**,

each container of files having its own unique identity associated therewith **(page 2, paragraph 22, line 9-11**, "...The core is primarily responsible for managing applications and their context as defined by the configuration files."; the application is identified by it's configuration files**)**,

~~*said identity comprising at least one of an IP address, a host name, and a Mac address;*~~

wherein, the plurality of files within each of the plurality of containers comprise one or more application programs including one or more processes, and associated system files for use in executing the one or more processes, each container having

its own execution file associated therewith for starting one or more applications **(implicit**, since it's a well-known design practise to have executable binaries or scripts to launch an application**)**,

in operation, each container utilizing a kernel resident on the server and wherein each container exclusively uses a kernel in an underlying operation system in which it is running and is absent its own kernel **(figure 1)***; and,*

a run time module for monitoring system calls from applications associated with one or more containers and for providing control of the one or more applications **(page 1, paragraph 4, line 9-15**, "...the system continually monitors the use of shared system resources..."**)**.

4.2 The **difference** between **D1** and the subject-matter of claim 1 is that the applications identity comprises one or more network parameters.

4.3 The **objective technical problem** to be solved by the present invention may therefore be regarded as providing an identifier which is unique within a global context.

4.4 The **solution** proposed cannot be considered as involving an inventive step since the incorporation of network parameters like hostname or IP address into an identifier is a common practise used to identify resources (including services) across global networks. The unified resource locator (URL) is widely used to identify processes as for example HTTP servers. Further known standards such as DCOM, CORBA, RPC and various Java technologies use network parameters in the identifiers. Therefore, it would be an obvious alternative to a person skilled in the art to include these parameters into a unique identifier.

4.5 Therefore, the subject-matter of claim 15 is **not inventive** with regard to Article 56 EPC.

5 Novelty of dependent claims

5.1 Dependent claims 2-3 do not appear to contain any additional features which, in combination with the features of any claim to which they refer, meet the requirements of the EPC with respect to **novelty**, the reasons being as follows (reference to the closest prior art is made in parentheses; the original wording of the claim is set in *italic font*):

5.2 **Claim 2**: an application in a container has no access to system files of applications in other containers **(D1, page 2, paragraph 19)**.

5.3 **Claim 3**: identifying all necessary files required for the application **(D1, page 6,**

paragraph 59-60; furthermore, it's obvious for a skilled person that in order to provide an individual execution environment, one of the main tasks is to identify all required components of an application**)** and providing an identifier for the application **(implicit, D1, page 2, paragraph 22)**.

6 Inventiveness of dependent claims

6.1 Dependent claims 4-14 and 16-17 do not appear to contain any additional features which, in combination with the features of any claim to which they refer, meet the requirements of the EPC with respect to **inventive step**, the reasons being as follows (reference to the closest prior art is made in parentheses; the original wording of the claim is set in *italic font*):

6.2 **Claim 4**: as set out previously in item 5.4, the inclusion of the hostname, the IP and MAC address, is an obvious alternative and therefore not inventive.

6.3 **Claim 5 and 7**: distributing various files across different servers within a network is a standard practise known to the person skilled in the art and therefore not inventive.

6.4 **Claim 6 and 17**: monitoring hardware resources is disclosed in **D1 (page 1, paragraph 4, line 13-15)**.

6.5 **Claim 8**: merely discloses an embodiment of an installation procedure without any surprising technical effect and is therefore not inventive.

6.6 **Claim 9-11**: **D1** discloses a graphical user interface to managed the application containers **(page 2, paragraph 22, line 1-6)**.

6.7 **Claim 12-14**: these claims merely specify standard tasks of an installation programm and are therefore not inventive but are rather in juxtaposition to the previous features.

6.8 **Claim 16**: monitoring resource usage and overriding system default settings is disclosed in **D1 (**see **page 1, paragraph 4, line 9-15)**.

7 Remarks to letter of reply, dated 31.03.2008

7.1 In his letter of reply, dated, the applicant claims that **D1** doesn't disclose a system where an application and system files are encapsulated from a local operating system kernel **(page 3, paragraph 3 - page 5, paragraph 1)** and that the encapsulated system files are used in place of the local ones. An example is given on **page 7, paragraph 4 - page 8, paragraph 1** where an application requires specific versions of shared libraries which are provided by the application container in order to make the application compatible with the kernel.
Contrary to all these statements, **D1** does disclose that a system file, MSVCRT.DLL,

is encapsulated within the abstraction and protection layer thus providing the application with the necessary version of a system file **(paragraph 13)**. This is exactly the example given in applicants letter of reply **(page 7, paragraph 4 - page 8, paragraph 1)**. MSVCRT.DLL is a operating system library containing the C library functions and is thus a vital part of the operating system (libc.so being the equivalent on a UNIX platform).

7.2 Furthermore, referring to the remarks on **page 12, paragraph 4, D1** is not limited to the Windows operating system. **D1** uses the windows operating system as one example embodiment **(paragraph 5, line 1-3**, "...for example, in embodiments within Windows-based operating systems..."**)** and discusses other operating systems as well **(**see for example **D1, paragraph 26, line 14-19**, "...On platforms other than Windows,...Macintosh has the System Folder, and other operating systems will have corresponding elements. It is important to note that on most UNIX systems..."**)**. The "pseudo-installation", the applicant is referring to on **page 12, paragraph 3** of his letter of reply, is a specific mode of operation which is clear when **paragraph 7** of **D1** is read completely and it discloses that the abstraction layer provides a *"...a virtual environment at the time the application runs..."* **(paragraph 7 , line 4-5)** and *"...dynamically changing the virtual environment according to administrative settings..."* **(paragraph 7, line 11-13)**.

7.3 In his letter of reply, the applicant claims that the unique identity of the container is an *"important feature"* which *"significantly"* distinguishes claim 1 from the prior art document **D1 (page 1, last paragraph - page 2, paragraph 1)**. Reading the letter of reply, it appears that an application ID and a container ID are the same thing since they both consist of IP address, host name, MAC address and system ID **(**see **page 2, paragraph 4**, for the definition of an application ID and **page 6, paragraph 1**, for the definition of a container ID). Therefore, a container ID is merely an application identifier, and is therefore **common practise** as already stated in the previous communication, dated 23.11.2007.

8 Conclusion

8.1 It is not at present apparent which part of the application could serve as a basis for a new, allowable claim. Should the applicant nevertheless regard some particular matter as patentable, an independent claim should be filed taking account of Rule 43(1) EPC. The applicant should also indicate how the subject-matter of the new claim differs from the state of the art and the significance thereof.

8.2 In case the applicant files a new set of claims, the applicant is requested to point out and discuss in his letter of reply any difference that would distinguish the subject-matter of the present application from what is disclosed in the available prior art. In particular, the applicant is requested to identify the technical problem that exists in the closest prior art, namely D1, describe how the applicant's invention solves this problem, and provide some argument for why this solution would not be obvious to a person skilled in the art.

8.3 When filing amended claims the applicant should at the same time bring the description into conformity with the amended claims. Care should be taken during revision, especially of the introductory portion and any statements of problem or advantage, not to add subject-matter which extends beyond the content of the application as originally filed (Article 123(2) EPC).

8.4 In order to facilitate the examination of the conformity of the amended application with the requirements of Article 123(2) EPC, the applicant is requested to clearly identify the amendments carried out, irrespective of whether they concern amendments by addition, replacement or deletion, and to indicate the passages of the application as filed on which these amendments are based.

Europäisches Patentamt
European Patent Office
Office européen des brevets

European Patent Office
80298 MUNICH
GERMANY
Tel: +49 89 2399 0
Fax: +49 89 2399 4465

Banzer, Hans-Jörg
Kraus & Weisert,
Thomas-Wimmer-Ring 15
80539 München
ALLEMAGNE

Formalities Officer
Name: Schmidt-Lorenz, S
Tel: +49 89 2399 - 7574
or call
+31 (0)70 340 45 00

Substantive Examiner
Name: Milasinovic, Goran
Tel: +49 89 2399 - 5611

Application No.	Ref.	Date
04 021 916.4 - 2211	14738EP /nh	03.06.2008
Applicant: Trigence Corp.		

Communication pursuant to Article 94(3) EPC

The examination of the above-identified application has revealed that it does not meet the requirements of the European Patent Convention for the reasons enclosed herewith. If the deficiencies indicated are not rectified the application may be refused pursuant to Article 97(2) EPC.

You are invited to file your observations and insofar as the deficiencies are such as to be rectifiable, to correct the indicated deficiencies within a period

of 4 months

from the notification of this communication, this period being computed in accordance with Rules 126(2) and 131(2) and (4) EPC.

One set of amendments to the description, claims and drawings is to be filed within the said period on separate sheets (R. 50(1) EPC).

Failure to comply with this invitation in due time will result in the application being deemed to be withdrawn (Art. 94(4) EPC).

Europäisches Patentamt • European Patent Office • Office européen des brevets •

Milasinovic, Goran
Primary Examiner
for the Examining Division

Europäisches Patentamt
European Patent Office
Office européen des brevets

European Patent Office
80298 MUNICH
GERMANY
Tel: +49 89 2399 0
Fax: +49 89 2399 4465

Enclosure(s): 7 page/s reasons (Form 2906)

- Front-End Munich PHB -

Non Scannable Objects - (Models.....)

File Number : 2211/0104

Directorate : 04021916.4

Date of receipt : EPO - Munich 86 31. März 2008

(n = amount)

			(For Supp. Desk only)
☐	(.....)n	Non Din A4 Document(s)	☐ b/w scanned / copied in A4
☒	(...7...)n	Coloured Document(s)	☐ b/w scanned / copied in A4
☐	(.....)n	Photograph(s)	☐ b/w scanned
☐	(.....)n	Prospect(s)/Brochure(s)	☐ b/w scanned
☐	(.....)n	Model(s)	
☐	(.....)n	Other :.....	

EPO - Munich 86
31. März 2008

Sent on : .....

To : J. Stewart - Room 1133 -

Models available from : M. Gazzoli / M. Jorba
- Support desk DG2 -

BD/WP/NO/NOSCA

03.04.08

Kraus & Weisert
Patent- und Rechtsanwälte
European Patent and Trademark Attorneys

EPO - Munich
86
31. März 2008

Kraus & Weisert · Thomas-Wimmer-Ring 15 · 80539 München

European Patent Office
Erhardtstraße 27
80469 Munich

Dr. Walter Kraus (– 2002)
Dr.-Ing. Annekäte Weisert (– 2002)
Dr. Thomas Albrecht
Dipl.-Ing. Hans-Jörg Banzer
Dr. Holger Adam
Dr. Inge Hiebl
Dr. Claus Beckmann
Dr. Florian Bertsch
Dr. Andreas Sticht
Dr. Horst Glaser
Dr. Markku Schwarz
Dr. Jörg Hoffmann
Dr. Ferdinand Nielsen
Rechtsanwalt

Unser Zeichen / Our Ref.: 14738EP HJB/AS/kr

Bitte in der Antwort angeben
Please refer to in your reply

March 31, 2008

Re: European Patent Application 04 021 916.4-2211
Trigence Corp.

Responsive to the Communication dated November 23, 2007.

I.

Regarding items 3.2 and 3.3 of the above-referenced Official Communication, the Request for Oral Proceedings pursuant to Art. 116 EPC in case the Examining Division intends to refuse the present application is maintained.

II.

In the above-referenced Official Communicaton, the Examining Division maintains the objections with regard to novelty and inventive step. However, in applicant's view the Examining Division has misread new claim 1, in particular the features added to claim 1 with the last response. While in the Official Communication it is stated that the new feature of claim 1 is "application software having its own unique identity", this is not what actually is recited in claim 1. Claim 1 defines that

each container of application software has its own unique identity, and that the application software has an identity of a container that it is associated with, which differs significantly from the feature identified in the above-referenced Official Communication.

Moreover, these features are indeed important for the capability of the instant invention to make it possible for applications to be executed on an operating system for which they were not intended or programmed.

In the following, the above and other features of the present invention distinguishing the instant invention over the prior art and the remarks in the above-referenced Official Communication will be discussed in more detail:

Remark 2.1

In the prior art, in order for a software application to execute correctly it must be combined with all of its dependencies including any third party elements and an operating system (OS) compatible with that software application. This prior art approach, results in a monolithic application stack, as shown in Figure 1, below. The elements in this prior art software stack are inseparable. There is no variance or the software application will not execute. Dependant system files and an application identity are embodied in the specific OS. An application identity consists of IP address, host name, MAC address, and system ID. These parameters are defined and managed in the OS. As a monolithic stack, an application executes in conjunction with the OS it is integrated with.

Installation of an application, particularly a desktop application, is something that is commonly performed with a high of degree of success by individuals without a specific software skill or background in information technology. Integration of a server application, particularly those server applications commonly used in an enterprise data center, with an OS is quite complex requiring a specific software skill set and an information technology background. Installation and integration of server applications require specific knowledge of the OS being used so that an application identity can be created.

Another distinguishing feature defined in claim 1 is that some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files resident on the server prior to said storing step so that executable applications can execute in conjunction with an operating system that said executable application is not originally intended to operate with and would otherwise not function properly, **and** wherein application software has an identity of a container that it is associated with.

It is all of these features that together allow an application to run under an operating system for which the application was not intended to run.

The invention defines that each container of application software, a unique application object, has its own unique identity.

No such containers of software are disclosed in any of the prior art.

The communication from the Examining Division also states that claim 1 is directed to the provision of compatibility between an application and an operating system and not to the generation of a unique identifier.

The applicant is completely confused by this statement.

As is clearly and succinctly illustrated above, the invention defined in the claims of the instant application provides secure containers of application software which each have a unique identifier and which each have the required files as defined in the claims, so as to be able to operate on different operating systems for which the application was not originally intended to run.

Specifically, *each container (20a-20c) of application software having its own unique identity and comprising one or more of the executable applications (21a-21f) for use with a local kernel (12) residing permanently on one of the servers.*

In accordance with this invention, Figure 3 illustrates that identity can now be embodied in a unique application object referred to as a secure application

Furthermore, D1 makes no disclsoure or allowance for system files, embodied in the OS to which the application has been integrated, to be included in the unique application object.

The disclosure of the instant invention teaches that system files that are embodied in a secure application container will be used by the application and any application dependencies, including for example, middleware. These system files are used by the application associated with the secure application container instead of system files embodied in the OS; wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files resident on the server prior to said storing step so that executable applications can execute in conjunction with an operating system that said executable application is not originally intended to operate with and would otherwise not function properly. This is the case even if the file in the OS is identified by the same name as the file in the application container.

An application embodied in a secure application container is able to execute correctly with an OS that the application was not originally integrated with. No change to the application or its dependencies is required where the application is embodied in an application container. With the state of prior art as well as the teaching within D1, in order for an application to execute correctly on an OS for which it was not originally intended to execute changes would be required in at least one or more of application executables, system files, configuration files, configuration syntax, directory structure, file system type, OS configuration of users, OS configuration of identity, OS configuration of authentication mechanisms, OS configuration of logging and error handling, OS configuration of network services, OS configuration of system services.

The reasons why an application embodied in a secure application container is able to execute correctly on an OS for which the application was not originally intended to operate stem from the ability to cause the application to utilize files and configuration elements embodied within the container in addition to system files in the OS. Where, for example, the application requires specific versions of shared libraries and the OS hosting the application does not supply the exact

identity. As such each secure container has an independent network identity. Moreover, the plurality of secure containers each have a unique identity in addition to the identity defined by the OS, as shown in figure 6. It is therefore clear that each secure container can be setup in a network environment. The ability for a secure container of application software to utilize its own unique network identity where that identity is independent of the OS is not defined in prior art nor in D1.

Remark 2.3

Under 2.3 the Examining Division states that D1 clearly discloses that applications written for a different version of the OS are adapted in order to be executable on the current OS (MSVCRT.DLL and OLE are both part of the OS (Para 31 & 36-37).

D1 discloses that if an application attempts to change a file in the OS, D1 uses the example of a shared library like MSVCRT.DLL, this change is localized to the application and the code resident on the OS, the client computer, is left unchanged. The examples and specific behavior disclosed by D1 assume and require that system files are the same as those found on the OS or client computer. D1 discloses that system files from the OS are embodied in the virtual application object. However, D1 does not disclose or teach a way in which an application in said virtual application object is able to execute correctly on a version of the OS for which the application was never intended to execute. The applicant defines in claim 1 required structures which allow this to occur. D1 discloses that system files are inclusive for the purpose of deploying an application without requiring an installation process. D1 uses the terms “pseudo-installation” and “installation-like” to describe this capability.

D1 discloses techniques that are specific to and limited to the Windows OS. The instant invention discloses techniques that are applicable to UNIX, Linux and Windows.

The instant invention discloses a capability that allows applications to utilize system files that are different from those found on the OS of the client or server

computer; more specifically and with reference to claim 1; wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files resident on the server prior to said storing step so that executable applications can execute in conjunction with an operating system that said executable application is not originally intended to operate with and would otherwise not function properly. This capability, as disclosed in the instant invention, is based on a technique called "spoofing". This technique is accomplished by means of system call intercepts and is disclosed in detail in the invention. This "spoofing" of system call behavior enables, among other capabilities, system files, shared libraries, configuration files, configuration syntax, and directory structures that are different from those of the OS on the client or server computer to be used by the application. Any translation that is needed to maintain compatibility between the application and the underlying kernel is performed as part of the "spoofing" process. This "spoofing" capability or any similar technique or concept for the purposes of allowing an application to operate with an OS that the application is not originally intended to operate with is not disclosed in D1 or prior art.

III.

In summary, it has been shown that the subject-matter of the new claims is both novel and inventive over the prior art. Therefore, in applicant's view the present application is ready for grant.

Dr. Andreas Sticht
European Patent Attorney

In order for an application to utilize an OS, other than the version for which it was originally intended to be integrated with, the software application would require significant modification.

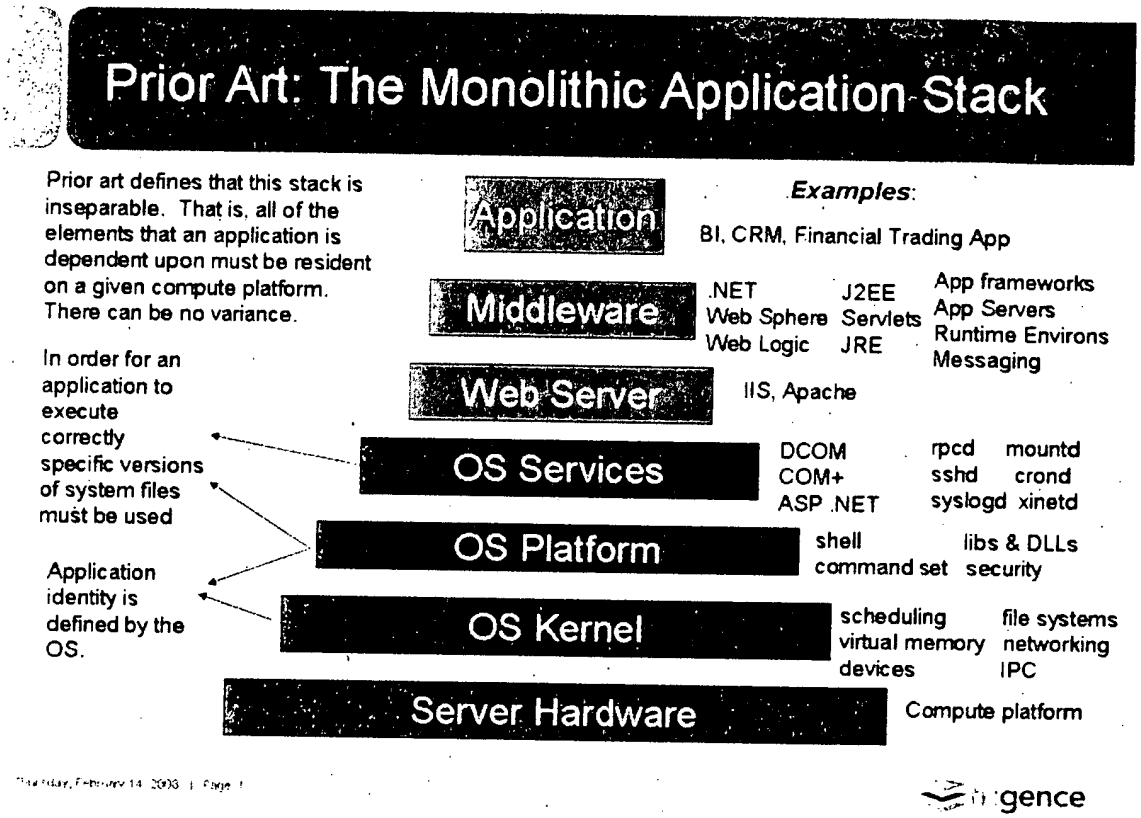


Figure 1

The invention taught in D1 takes a first step at separating the monolithic application stack. D1 teaches that an application can be distinct from the monolithic application stack of prior art. This is shown in Figure 2.

However D1 does not teach or suggest that system files, part of an OS, can be encapsulated with a software application as part of an application .object. Furthermore, D1 does not teach that an identity can be defined as part of an application specific object; and D1 does not teach that an application is able to execute correctly on an OS in which it was never intended to operate with.

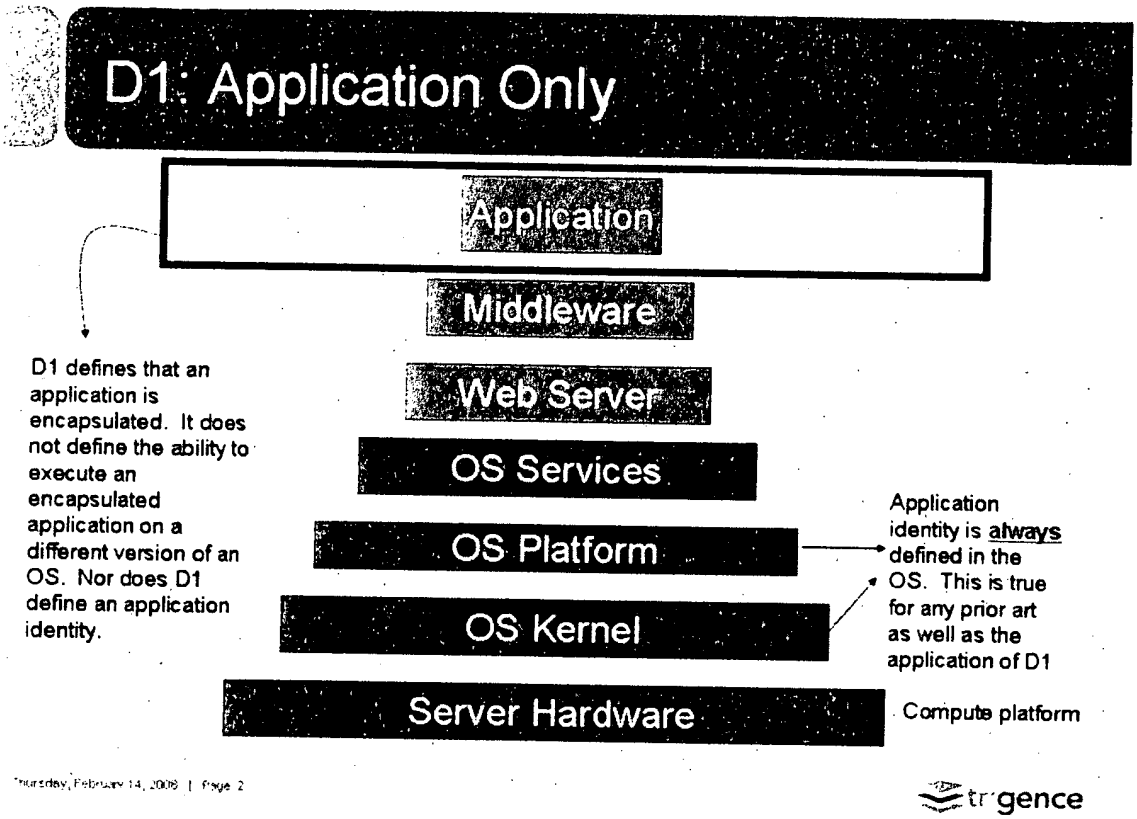


Figure 2

In the Communication from the Examining Division it is said that it is common practice for application software to have its own unique identity.

With all due respect, the applicant would like to point out that the applicant's claim does not recite application software having its own unique identity. The applicant's invention defines the step of providing secure containers of application software *wherein each container has a unique identity*. Furthermore, the secure containers of application software comprise one or more of the executable applications and a set of associated system files required to execute the one or more executable applications for use with a local kernel residing permanently on one of the servers.

The applicant's claim further defines that the aforementioned set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems. Yet still further, the containers (20a-20c) of application software in claim 1 are said to exclude a kernel.

container. The invention discloses and claims that an identity can exist in the OS in addition to an identity embodied in the secure application container. Moreover, multiple secure application containers can be hosted on a single OS and each container can have its own unique identity. A container identity is defined by the invention e.g. as one or more of IP address, host name, MAC address, and system ID.

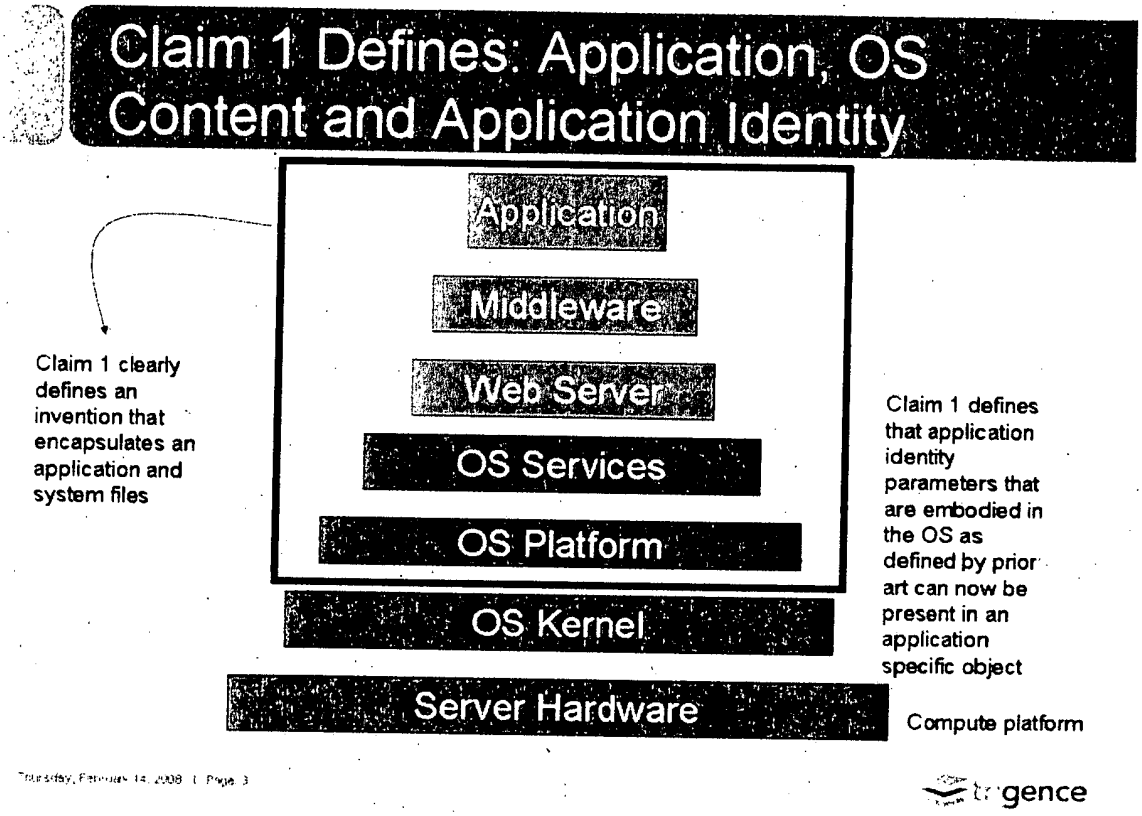


Figure 3

In summary, this invention defines within the claims that executable applications in addition to specific system files are embodied in a secure application container; wherein the set of associated system files are compatible with a local kernel (12) of at least some of the plurality of different operating systems, the containers (20a-20c) of application software excluding a kernel. Whereas prior art teaches that a unique application object is a monolithic stack including application executables, application dependencies and the specific OS to which the application has been integrated.

version of a required shared library the application would not execute correctly. The application would need to be re-integrated for use with the OS hosting the application in this case. Where the same application is deployed within an application container on an OS that supplies an incompatible version of shared libraries the application will execute correctly, with out change, because the application will use shared libraries embodied in the secure application container in lieu of the shared libraries supplied by the OS. The same example applies to at least, but is not limited to, variances in specific configuration syntax, specific location of files, specific directory structure and content.

The inventive aspects of this invention are significant when viewed in the context of an enterprise data center. Consider that today; driven by the monolithic application stack, the data center is dominated by what is referred to as "Islands of Compute", as shown in Figure 4. This results in server resources being dedicated to specific application related tasks without much flexibility in re-purposing a server or performing maintenance.

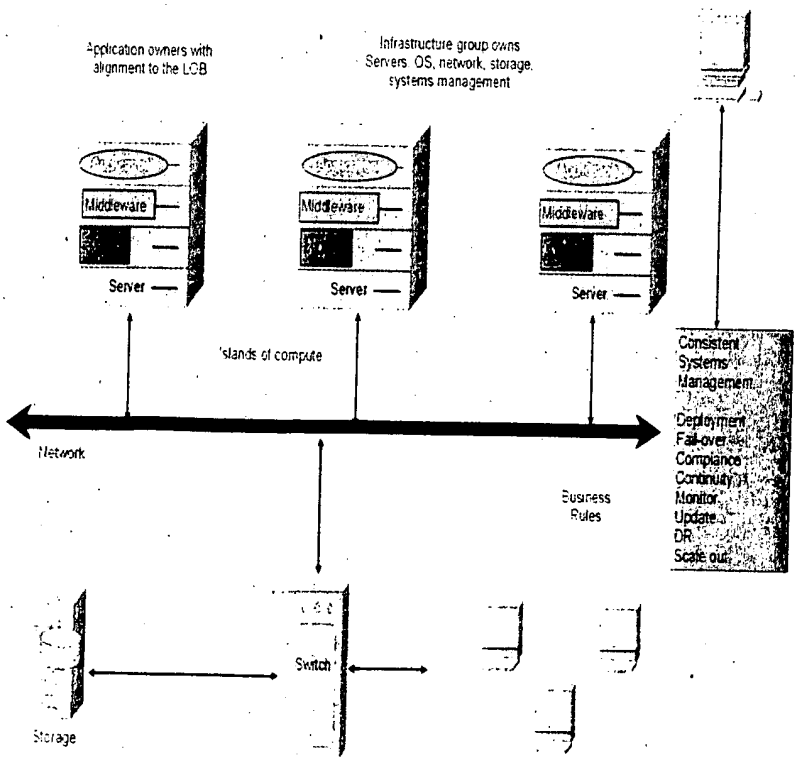


Figure 4

The architects of the enterprise data centers express interest in a design based on a pool of common servers, as shown in Figure 5, where any application can be placed on any server at any time. The flexibility in this design approach is significant when compared to the current state of the art. It is considered by architects and researchers alike to be extremely difficult to realize the desired data center design based on the state of prior art as it relates to application deployment and the monolithic application stack.

Any Application on Any Server at Any Time

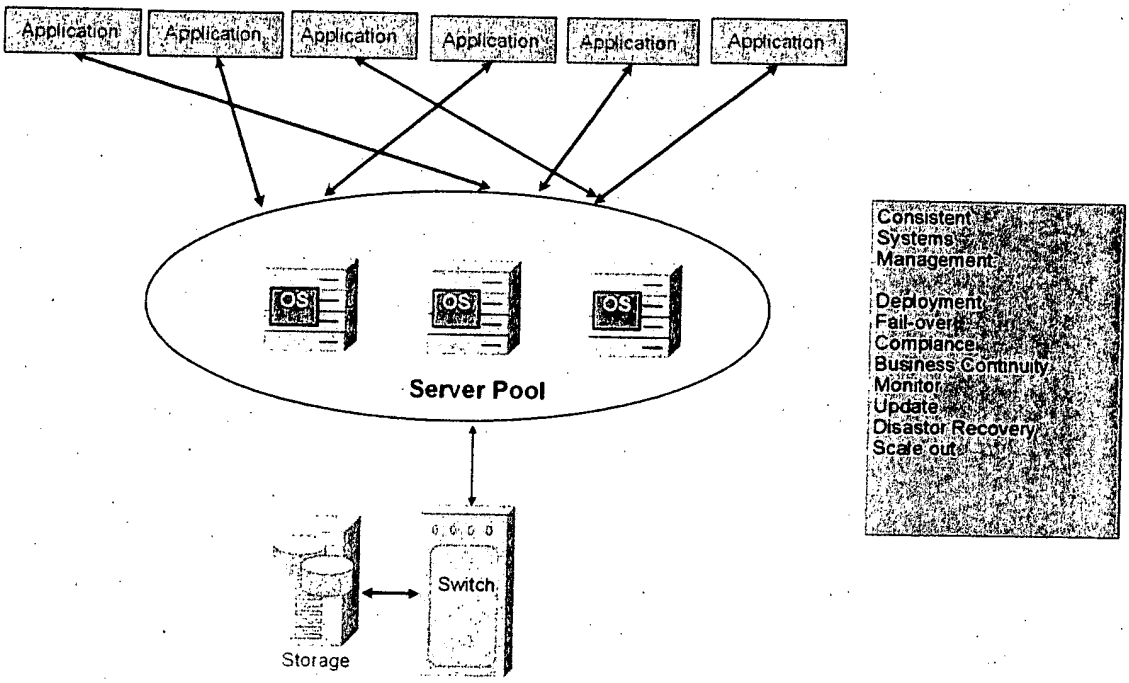


Figure 5

A level of hardware consolidation is possible with technology described as a virtual machine monitor. It allows multiple application images to co-exist on the same server. Figure 6 illustrates the consolidation of monolithic application stacks on a server platform. However, this approach does not alleviate the need for a monolithic application stack.

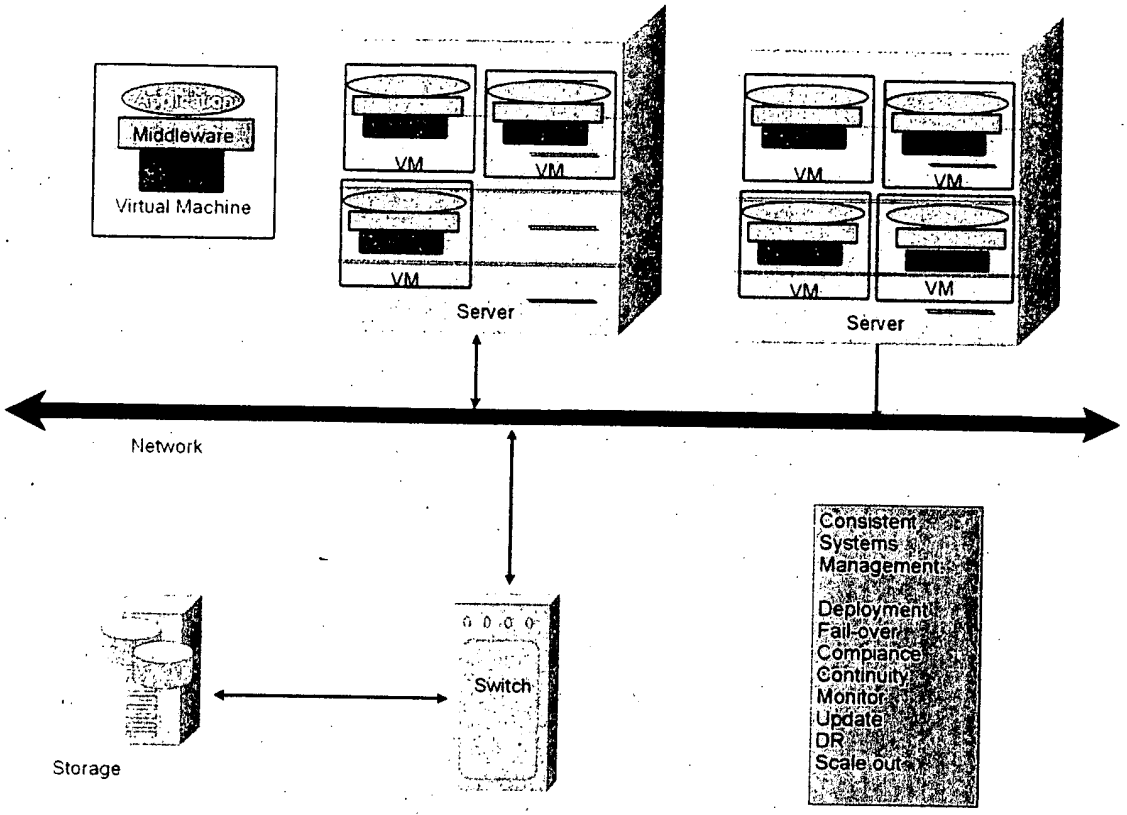


Figure 6

The implementation of secure application containers where; each container (20a-20c) of application software having its own unique identity and comprising one or more of the executable applications (21a-21f) and a set of associated system files required to execute the one or more executable applications (21a-21f), for use with a local kernel (12) residing permanently on one of the servers; wherein the set of associated system files are compatible with a local kernel (12) of at least some of the plurality of different operating systems, the containers (20a-20c) of application software excluding a kernel, and wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files resident on the server prior to said storing step so that executable applications can execute in conjunction with an operating system that said executable application is not originally intended to operate with and would otherwise not function properly, and wherein application software (21a-21f) has an identity of a container that it is associated with enables a level of flexibility and automation in the enterprise data center that has never before been possible. Figure 7 demonstrates how architects of the enterprise data center are using secure application containers to deliver their next generation designs.

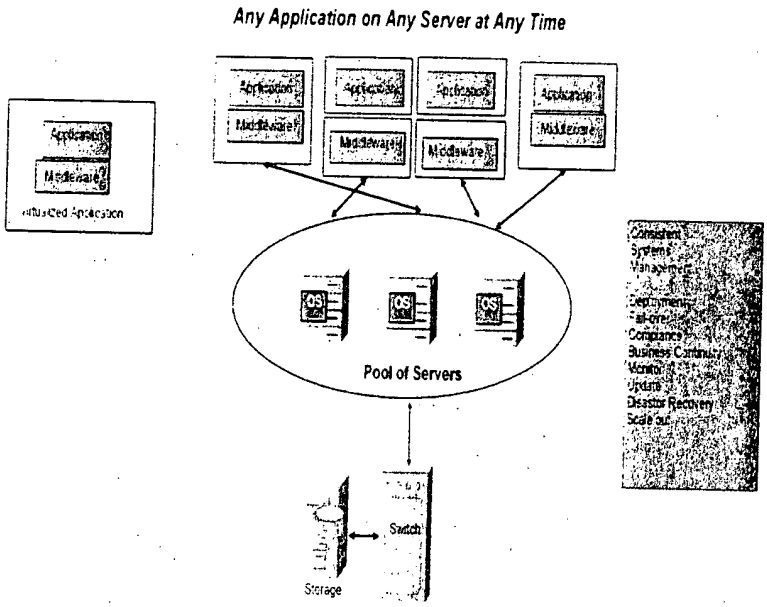


Figure 7

The data center as envisioned by enterprise architects is realized in large part due to the ability to manage an application and all or part of its dependencies as a unique object.

Remark 2.2

Under 2.2 the Examining Division states that: D1 discloses only one server. However, it is obvious that more of these systems as depicted in D1 figure 1 can be setup in a network environment. Furthermore, the servers depicted in claim 1 have no interaction defined between each other and therefore appear to operate independently/juxtaposed.

Claim 1 defines a plurality of secure containers of application software, **each container of application software having its own unique identity.** The unique identity of each container of application software is defined e.g. as one or more of IP address, MAC address, hostname and system ID. These parameters that make up the unique identity defined in claim 1 are the parameters used to allow a service to be identified in a network environment. For example, a MAC address and an IP address must be defined in order for a service to be usable in an IP network. It is therefore clear that the unique identity defined in claim 1 is a network

The examination is being carried out on the **following application documents**:

Claims, Numbers

1-17	received on	18.08.2006	with letter of	18.08.2006

Drawings, Sheets

1/17-13/17, 15/17-17/17	as originally filed			
14/17	received on	09.11.2004	with letter of	09.11.2004

1 Summary

1.1 All objections with regard to novelty and inventive step, as set out in communication, dated 05.05.2006, are **maintained**.

2 Remarks to letter of reply, dated 18.08.2006

2.1 The new feature of claim 1, *"application software having its own unique identity"* is common practise in system design and therefore implicit since each entity in a computing system (object, device, file, etc.) has it's own identifier in order to be accessed. If this is in the form of an address pointer (object), a moniker (common object model reference), a filename or an IP address for a node on the network is irrelevant since it is merely a design choice which mainly depends on the underlying specification. Furthermore, claim 1 is directed to the provision of compatibility between an application and an operating system and not to the generation of a unique identifier. Therefore, **these features are juxtaposed** and cannot provide an **inventive step** as required by Article 56 EPC.

2.2 **D1** discloses only **one server**. However, it is obvious that more of these systems, as depicted in **D1, figure 1**, can be setup in a network environment. Furthermore, the servers specified in claim 1 have no **interaction defined between each other** and

therefore appear to operate independently/juxtaposed.

2.3 **D1** clearly discloses that applications which were written for a different version of the operating system are adapted in order to be executable on the current operating system version **(paragraph 31 and 36-37**; MSVCRT.DLL and OLE are **both part of the operating system)**.

3 Conclusion

3.1 At least some of the objections raised above are such that there appears to be no possibility of overcoming them by amendment. **Refusal of the application under Article 97(1) EPC is therefore to be expected.**

3.2 The examining division considers that the case has been clarified to an extent that a decision could be reached now **without oral proceedings**.

3.3 Considering that the EPO aims in the interest of the public to bring the proceeding to a conclusion as soon as possible, and to avoid unnecessary costs, the applicant is invited to declare withing the given time limit, whether in view of the provisional conclusion put forward hereinbefore, the request for oral proceeding is maintained.

3.4 Applicant's attention is also drawn to the fact that the applicant may request for a **decision according to the state of the file**. Such a decision is appealable as well as any other kind of decision.

3.5 In the further procedure, **Article 123(1) EPC and Rule 86(3) EPC will be strictly applied**: Admissibility of further new sets of claims under Rule 86(3) EPC will be subject to the compliance, "prima facie", to Rule 29(2),(5) EPC, to Article 123(2) EPC, Article 84 EPC and finally to Rules 29(1)/27(1)(b) EPC.

EPA/EPO/OEB
D-80298 München

+49 89 2399-0
TX 523 656 epmu d
FAX +49 89 2399-4465

Europäisches Patentamt
Generaldirektion 2

European Patent Office
Directorate General 2

Office européen des brevets
Direction Générale 2

Banzer, Hans-Jörg
Kraus & Weisert,
Thomas-Wimmer-Ring 15
80539 München
ALLEMAGNE

Telephone numbers:

Primary Examiner (substantive examination) +49 89 2399-5611

Formalities Officer / Assistant (Formalities and other matters) +49 89 2399-7574

Application No.	Ref.	Date
04 021 916.4 - 2211	14738EP /nh	23.11.2007
Applicant: Trigence Corp.		

Communication pursuant to Article 96(2) EPC

The examination of the above-identified application has revealed that it does not meet the requirements of the European Patent Convention for the reasons enclosed herewith. If the deficiencies indicated are not rectified the application may be refused pursuant to Article 97(1) EPC.

You are invited to file your observations and insofar as the deficiencies are such as to be rectifiable, to correct the indicated deficiencies within a period

of 4 months

from the notification of this communication, this period being computed in accordance with Rules 78(2) and 83(2) and (4) EPC.

One set of amendments to the description, claims and drawings is to be filed within the said period on separate sheets (Rule 36(1) EPC).

Failure to comply with this invitation in due time will result in the application being deemed to be withdrawn (Article 96(3) EPC).

Milasinovic, Goran
Primary Examiner
for the Examining Division

Enclosure(s): 2 page/s reasons (Form 2906)

Registered Letter
EPO Form 2001 07.02CSX

Europäisches Patentamt
GD2

European Patent Office
DG2

Office européen des brevets
DG2

04021916.4 - 2211

SEPU: 19.10.05
PACT:

16.11.06

Loss of particular rights

DEST/ **AT** 07.08.06 (=LOPR(2))
DEST/ **BE** 07.08.06 (=LOPR(2))
DEST/ **BG** 07.08.06 (=LOPR(2))
DEST/ **CH** 07.08.06 (=LOPR(2))
DEST/ **CY** 07.08.06 (=LOPR(2))
DEST/ **CZ** 07.08.06 (=LOPR(2))
DEST/ **DK** 07.08.06 (=LOPR(2))
DEST/ **EE** 07.08.06 (=LOPR(2))
DEST/ **ES** 07.08.06 (=LOPR(2))
DEST/ **FI** 07.08.06 (=LOPR(2))
DEST/ **GR** 07.08.06 (=LOPR(2))
DEST/ **HU** 07.08.06 (=LOPR(2))
DEST/ **IE** 07.08.06 (=LOPR(2))
DEST/ **IT** 07.08.06 (=LOPR(2))
DEST/ **LU** 07.08.06 (=LOPR(2))
DEST/ **MC** 07.08.06 (=LOPR(2))
DEST/ **NL** 07.08.06 (=LOPR(2))
DEST/ **PL** 07.08.06 (=LOPR(2))
DEST/ **PT** 07.08.06 (=LOPR(2))
DEST/ **RO** 07.08.06 (=LOPR(2))
DEST/ **SE** 07.08.06 (=LOPR(2))
DEST/ **SI** 07.08.06 (=LOPR(2))
DEST/ **SK** 07.08.06 (=LOPR(2))
DEST/ **TR** 07.08.06 (=LOPR(2))

Date of Legal Eff: 20.4.06

1. The time limit under Rule 69(2) EPC has expired. No appeal or application under Article 122 or Rule 69(2) EPC has been filed.
   The loss of particular rights has become final.

2. Loss of particular rights and coding that finding has become final (LOPR 3).

16-11-2006
. . . . . .
Date

Lambert, Karine
. . . . . .
Formalities Officer

Claims:

1. In a system having a plurality of servers (10a, 10b) with operating systems that differ, operating in disparate computing environments, wherein each server includes a processor (13) and an operating system including a kernel (12), a set of associated local system files compatible with the processor (13), a method of providing at least some of the servers in the system with secure, executable, applications (21a-21f) related to a service, wherein the executable applications (21a-21f) may be executed in a secure environment, wherein the executable applications (21a-21f) each include an object executable by at least some of the different operating systems for performing a task related to the service, the method comprising the steps of:

storing in memory accessible to at least some of the servers a plurality of secure containers (20a-20c) of application software (21a-21f), each container (20a-20c) of application software having its own unique identity and comprising one or more of the executable applications (21a-21f) and a set of associated system files required to execute the one or more executable applications (21a-21f), for use with a local kernel (12) residing permanently on one of the servers; wherein the set of associated system files are compatible with a local kernel (12) of at least some of the plurality of different operating systems, the containers (20a-20c) of application software excluding a kernel, and wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files resident on the server prior to said storing step so that executable applications can execute in conjunction with an operating system that said executable application is not originally intended to operate with and would otherwise not function properly, and wherein application software (21a-21f) has an identity of a container that it is associated with.

2. A method as defined in claim 1, wherein each container (20a-20c) has an execution file associated therewith for starting the one or more applications (21a-21f), and wherein the execution file includes instructions related to an order in which executable applications

within will be executed, and wherein in operation when applications are executed, applications within a container (20a-20c) have no access to system files or applications in other containers (20a-20c) or system files within the operating system during execution thereof if those applications or system files are read/write files.

3. A method as defined in claim 1 or claim 2 further comprising the step of pre-identifying applications and system files required for association with the one or more containers (20a-20c) prior to said storing step, and modifying at least some of the system files to provide an association with a container specific identity assigned to the container.

4. A method as defined in any one of claims 1, 2, or 3 comprising the step of assigning said unique associated identity to each of a plurality of the containers (20a-20c), wherein the identity includes at least one of IP address, host name, and MAC address, a unique identifier of a network node.

5. A method as defined in any one of claims 1 through 4, wherein the one or more applications (21a-21f) and associated system files are retrieved from a computer system having a plurality of secure containers (20a-20c).

6. A method as defined in any one of claims 2 through 5 wherein server information related to hardware resource usage including at least one of CPU memory, network bandwidth and disk allocation is associated with at least some of the containers (20a-20c) prior to the applications (21a-21f) within the containers (20a-20c) being executed.

7. A method as defined in any of claims 2 through 6, wherein containers include files stored in network file storage (82), and parameters forming descriptors of containers stored in a separate location (84).

8. A method as defined in any of claims 1 through 7 comprising the step of creating containers (20a-20c) prior to said step of storing containers (20a-20c) in memory, wherein the step of creating containers comprises the steps of:

a) running an instance of a service on a server,

determining which files are being used, and,

copying applications and associated system files to memory;

or,

(b) using a skeleton set of system files as a container starting point and installing applications into that set of files.

9. A method as defined in any of claims 1 to 8 further comprising the step of installing a service on a target server selected from one of the plurality of servers (10a, 10b), wherein the step of installing the service includes the steps of:

using a graphical user interface, associating a unique icon representing a service with an unique icon representing a server for hosting applications related to the service and for executing the service, so as to cause the applications to be distributed to, and installed on the target server.

10. A method as defined in claim 9 wherein the target server and the graphical user interface are at remote locations, or wherein the graphical user interface is installed on a computing platform, and wherein the computing platform is a different computing platform than the target server.

11. A method as defined in any of claims 9 or 10, wherein the step of associating includes the step of relatively moving the unique icon representing the service to the unique icon representing a server.

12. A method as defined in any of claims 9 through 11 further comprising the step of: de-installing a service from a server, comprising the steps of:

displaying the unique icon representing the service;

displaying the unique server icon representing the server on which the service is installed;

and utilizing the icon representing the service and the icon representing the server to initiating the de-installation of the selected software application from the selected server.

13. A method according to any of claims 9 through 12 further comprising the step of separating the icon representing the service from the icon representing the server.

14. A method according to claim 13 further comprising the step of copying data file changes specific to the service back to a storage medium from which the data file changes originated prior to installation.

15. A computing system for performing a plurality of tasks each comprising a plurality of processes comprising:

a plurality of secure stored containers (20a-20c) of associated files accessible to, and for execution on, one or more servers (10a, 10b), each container (20a-20c) being mutually exclusive of the other, such that read/write files within a container (20a-20c) cannot be shared with other containers, each container (20a-20c) of files having its own unique identity associated therewith, said identity comprising at least one of an Internet Protocol address, a host name, and a MAC address, a unique identifier of a network node; wherein, the plurality of files within each of the plurality of containers comprise one or more application programs (21a-21f) including one or more processes, and associated system files for use in executing the one or more processes, each container (20a-20c) having its own execution file associated therewith for starting one or more applications, in operation, each container (20a-20c) utilizing a kernel (12) resident on the server (10a, 10b) and wherein each container (20a-20c) exclusively uses a kernel (12) in an underlying operation system in which it is running and is absent its own kernel; and,

a run time module for monitoring system calls from applications associated with one or more containers and for providing control of the one or more applications.

16. A computing system as defined in claim 15, further comprising a scheduler comprising values related to an allotted time in which processes within a container (20a-20c) may utilize predetermined resources and, wherein the run time module includes an intercepting module associated with the plurality of containers (20a-20c) for intercepting system calls from any of the plurality of containers (20a-20c) and for providing values alternate to values the kernel

would have assigned in response to the system calls, so that the containers can run independently of one another without contention, in a secure manner, the values corresponding to at least one of the Internet Protocol address, the host name and the MAC address.

17. A computing system as defined in claim 16, wherein the run time module performs:

monitoring resource usage of applications executing;

intercepting system calls to kernel mode, made by the at least one respective application within a container, from user mode to kernel mode;

comparing the monitored resource usage of the at least one respective application with the resource limits; and,

forwarding the system calls to a kernel on the basis of the comparison between the monitored resource usage and the resource limits.

deployed. Such deployment makes it quite costly to manage the number of systems required to support several applications.

There are existing solutions that address the single use nature of computer systems. These solutions each have limitations, some of which this invention will address. Virtual Machine technology, pioneered by VmWare, offers the ability for multiple application/operating system images to effectively co-exist on a single compute platform. The key difference between the Virtual Machine approach and the approach described herein is that in the former an operating system, including files and a kernel, must be deployed for each application while the latter only requires one operating system regardless of the number of application containers deployed. The Virtual Machine approach imposes significant performance overhead. Moreover, it does nothing to alleviate the requirement that an operating system must be licensed, managed and maintained for each application. The invention described herein offers the ability for applications to more effectively share a common compute platform, and also allow applications to be easily moved between platforms, without the requirement for a separate and distinct operating system for each application.

A product offered by Softricity, called SoftGrid®, offers what is described as Application Virtualization. This product provides a degree of separation of an application from the underlying operating system. Unlike Virtual Machine technology a separate operating system is not required for each application. The SoftGrid® product does not isolate applications into distinct environments. Applications executing within a SoftGrid® environment don't possess a unique identity.

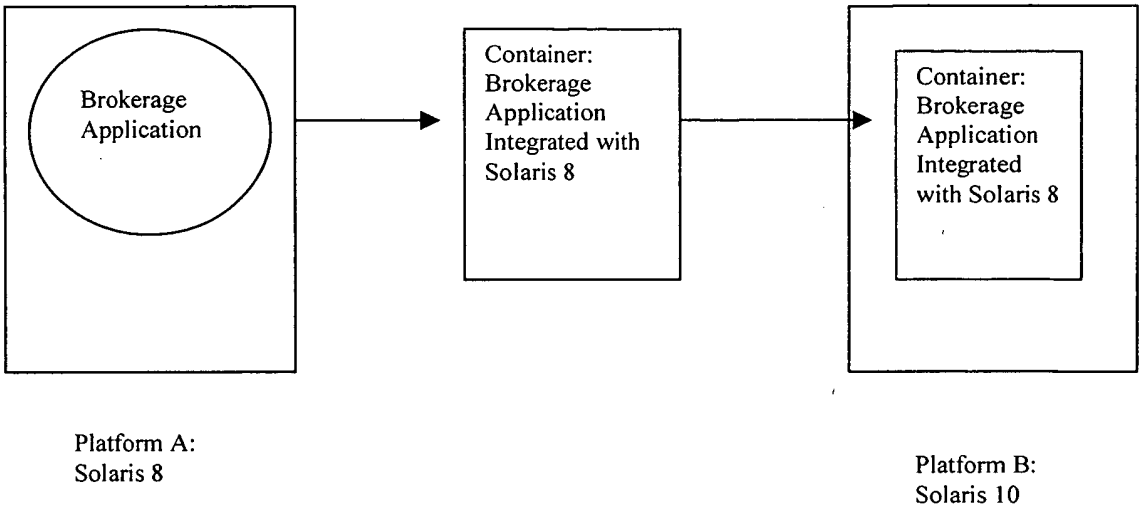
In US 2002/0174215 A1 an operating system abstraction and protection layer is disclosed. Through this layer, it is possible to run applications on a computer without changing the operating system of the computer, for example by modifying or adding system files. To achieve this, changes which would normally be made directly to the operating system are selectively made within the context of the application and the abstraction and protection

layer. The system disclosed in this document enables to run applications on the fly without a complex installation.

This invention provides a solution whereby a plurality of services can conveniently be installed on one or more servers in a cost effective and secure manner.

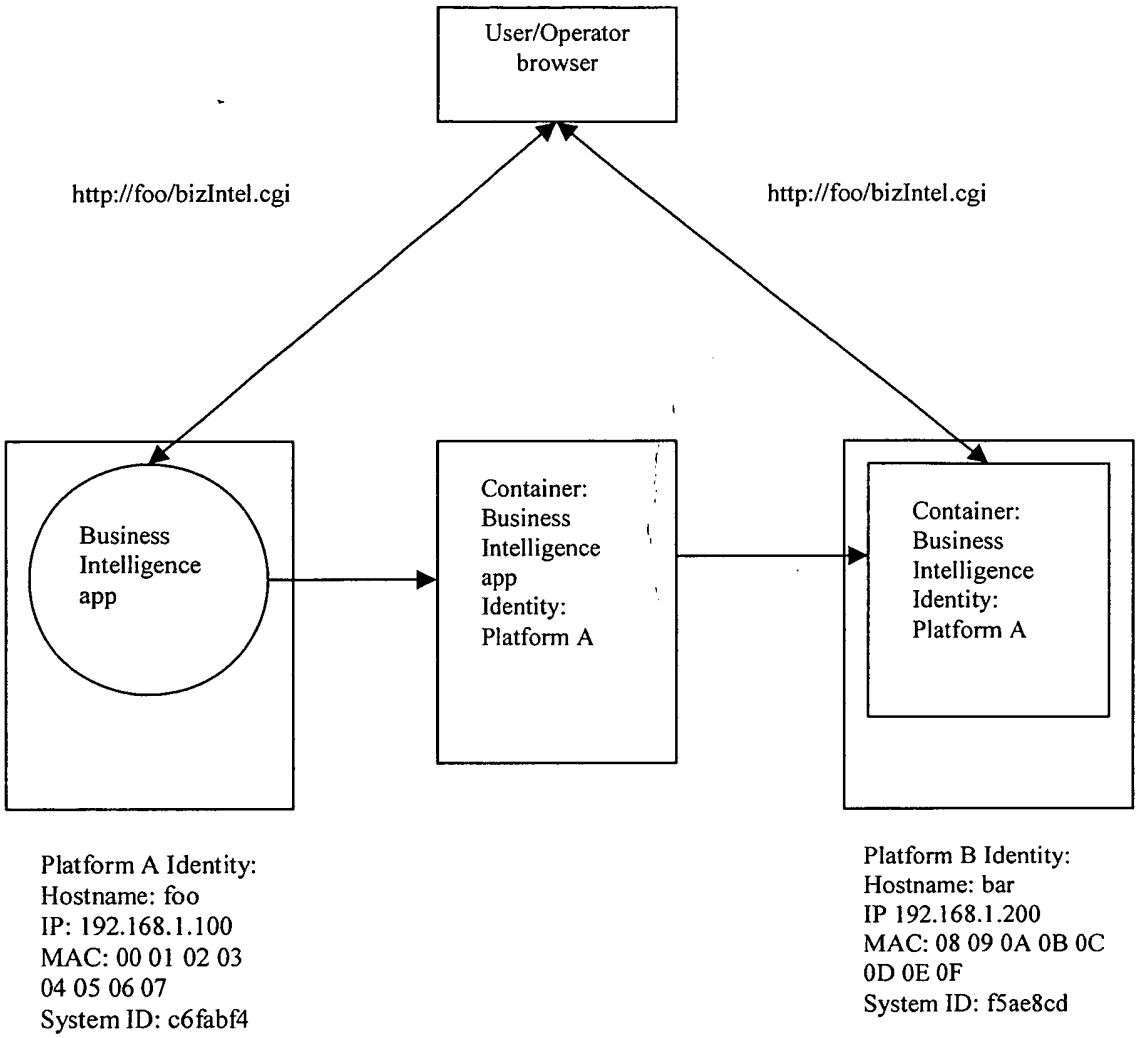
The following definitions are used herein:

Brokerage Application Example:



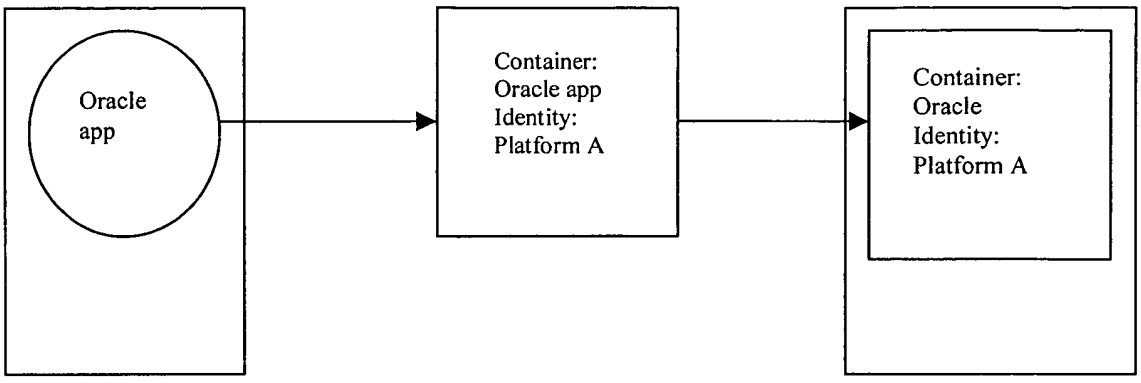
Exemplary Figure 1

Business Intelligence Application example



Exemplary Figure 2

Oracle Application Example



Platform A Identity:
Hostname: foo
IP: 192.168.1.100
MAC: 00 01 02 03 04 05 06 07
System ID: c6fabf4

Platform B Identity:
Hostname: bar
IP 192.168.1.200
MAC: 08 09 0A 0B 0C 0D 0E 0F
System ID: f5ae8cd

Exemplary Figure 3

Kraus & Weisert
Patent- und Rechtsanwälte
European Patent and Trademark Attorneys

Kraus & Weisert · Thomas-Wimmer-Ring 15 · 80539 München

European Patent Office
Erhardtstraße 27
80469 Munich

Dr. Walter Kraus (– 2002)
Dr.-Ing. Annekäte Weisert (– 2002)
Dr. Thomas Albrecht
Dipl.-Ing. Hans-Jörg Banzer
Dr. Holger Adam
Dr. Inge Hiebl
Dr. Claus Beckmann
Dr. Florian Bertsch
Dr. Andreas Sticht
Dr. Ferdinand Nielsen
Rechtsanwalt

Unser Zeichen / Our Ref.: 14738EP HJB/AS/bp

Bitte in der Antwort angeben
Please refer to in your reply

August 18, 2006

Re: European Patent Application 04 021 916.4-2211
Trigence Corp.

Responsive to the Communication dated May 5, 2006.

I.

Enclosed new claims 1-17 are filed intended to replace present claims 1-17.

New claim 1 is based on present claim 1. The following amendments have been performed:

- The term "applications" has been replaced by "executable applications" for reasons of conciseness. That the applications are executable is already disclosed in line 5 of present claim 1 and it is also clear from line 12 of claim 1, where it is stated that the associated system files are required to execute the applications, which means that the applications are executable.

- It has been added that each container of application software has its own unique identity, and that the application software has an identity of a container that it is

associated with. The unique identity of each container is for example disclosed in present claim 4. That the applications within the container share the identity of the container may be seen from page 5, lines 23-31, page 12, lines 4-11 and page 13, lines 16-18 of the application as filed.

- Finally, it has been clarified that the present application enables the applications to execute in conjunction with an operating system that said executable applications is not originally intended to operate with and would otherwise not function properly. This amendment is already apparent from present claim 1 where in the first lines it is stated that the invention relates to a system with a plurality of service with operating systems that differ and wherein the applications each include an object executable by at least some of the different operating systems. Furthermore, already in the introduction of the specification on page 1, lines 23-25 it is stated that in general the present invention relates to situations wherein certain applications require specific versions of operating systems.

New claims 2-17 correspond to present claims 2-17, wherein in claims 4, 15 and 16 the amendments proposed by the Examining Division under item 3 of the Official Communication have been performed in order to overcome the clarity objections.

Reference numerals have been added to all the claims.

In contrast to the Examining Division's proposal, independent claims 1 and 15 have been drafted in one-part form. As will be shown below, document D1 cited in the Official Communication relates to a totally different problem and therefore, document D1 is generally not suitable as closest prior art. In contrast, the present invention provides a method and a system addressing a different problem than D1 and providing a unique solution to this problem which is not known from the prior art.

II.

Furthermore, new description pages 2, 2a are filed replacing present description page 2. With the new description pages, a short appreciation of prior art document D1 has been added to the description. In applicant's view, no further modifications are necessary to bring the description in conformity with the new claims.

III.

In the above referenced Official Communication, the Examining Division is of the opinion that present claim 1 is not novel or at least not inventive over prior art document D1. New claim 1 has been clarified to reflect the differences between document D1 and the present invention more precisely.

In particular, the present invention as defined by new claim 1 differs from the system and methods disclosed in document D1 by the following features:

- According to new claim 1, a unique identifier identifies each container and the application associated therewith. No such unique identifier for both the container and the applications associated therewith is disclosed in document D1. Instead, D1 merely states that the core 102 is primarily responsible for managing applications and their context as defined by the configuration files. This passage merely means that the configuration files contain the information for managing the applications, but does not make any statement about how the applications are identified. In particular, each application may be attributed some task number or the like as usually in multitasking systems by the operating system. No hint is given in document D1 that applications and their container share a unique identity.

- Furthermore, document D1 only relates to a system having a single operating system (reference numeral 10). It does not, however, relate to a system having a plurality of servers with operating systems that differ, wherein the executable applications each include an object executable by at least some of the different operating systems as defined in claim 1.

That these features are implicit in D1 as stated in the Official Communication cannot be followed, since a feature is disclosed only implicitly if it is necessarily present in the teaching of a document. Since document D1 only relates to a single computer having an operating system, the features mentioned above are not necessarily present in D1 and therefore not implicitly disclosed (see Guidelines C-IV, 7.5).

For the above reasons, the subject-matter as defined by new claim 1 is novel over prior art document D1.

IV.

However, new claim 1 is not only novel, but also involves the necessary inventive step.

As can be seen from the first lines of new claim 1, it is an object of the present invention to provide a method of providing services in a system with different operating systems with secure executable applications. In other words, it is an object of the present invention to provide a method for handling applications in a system wherein different servers have different operating systems or even otherwise different properties.

This object is achieved by a method according to claim 1.

In particular, the method as defined in claim 1 provides the possibility of running applications on different operating systems even if the respective application was not designed to run on a specific operating system originally. In order to achieve this, the present invention uses containers, wherein executable applications are associated with containers and share a unique identity with these containers so they may be attributed to the correct container without problems. Furthermore, as defined in claim 1, the container comprises a set of associated system files required to execute the applications, wherein the associated system files are compatible with a local kernel of at least some of the plurality of different operating

systems and the associated system files are utilized in place of the associated local system files. In other words, for being able to run the application on an operating system it is not originally intended to be used with, the local system files of the operating system are replaced with the associated system files of the container, and only the kernel of the system is used.

In contrast thereto, document D1 relates to a completely different problem. In particular, document D1 relates to the problem of being able to run applications on a specific system on the fly without complex installation processes. As explained in section [0003] of D1, usually when software is installed on an operating system, system files are modified or added depending on the software to be installed. When uninstalling the software, these changes have to be reversed, which is often difficult to do since multiple installed applications or software packages may modify the same system file and therefore it is difficult to reverse exactly those changes which were caused by the software to be deinstalled.

To solve this, document D1 provides an operating system guard (see abstract of D1) which provides, for each application, more or less duplicates of the system files of the operating system, and only these duplicates are modified. Therefore, the operating system itself remains unchanged, and each application sees its own modified system files (see sections [0004]-[0007] of D1). However, document D1 is not concerned at all with running applications on different operating systems. Therefore, document D1 cannot give a person skilled in the art any hint how to realize a method which enables applications to run on different operating systems.

To make this difference even clearer, the mechanism of the present invention will be further explained below using two examples.

A first example is shown in the attached Exemplary Figure 1. In this figure, a brokerage application is shown that is used for example by a financial services company which is executing under a Solaris 8 operating system environment. As defined in new claim 1, the brokerage application is placed in a container, the container including the brokerage application itself as well as the associated system

files required to execute the brokerage application. These system files may for example include Solaris 8 libraries and the like.

Therefore, when the container is placed on a Solaris 10 platform, i.e. a different operating system, as shown on the right side of Exemplary Figure 1, in the container environment the brokerage application executes using the associated system files like Solaris 8 libraries on the Solaris 10 platform. No changes are required to the application or the Solaris 8 platform in order to enable the application to execute on the Solaris 10 platform.

Without the method of the invention as defined in new claim 1, the brokerage application from Solaris 8 would not successfully execute on the Solaris 10 platform for at least the following reasons:

a) Solaris 8 libraries are not fully compatible with Solaris 10;

b) the brokerage application is not integrated with Solaris 10 libraries, sufficient differences exist that will cause changes in the brokerage application at some level;

c) system call differences exist between Solaris 8 and Solaris 10;

d) the location of certain configuration files will be different in Solaris 10 as compared to Solaris 8; as such the application and/or the operating system will need to change in order for the brokerage application to function properly on a Solaris 10 platform;

e) the syntax of certain configuration files are different between Solaris 8 and Solaris 10; because of this normally the application and/or the operating system will have to be changed in order for the application to function properly on the Solaris 10 platform;

f) the brokerage application would need to be redesigned and fully integrated with Solaris 10 in order to operate effectively.

Furthermore, in accordance with new claim 1, as explained above a unique identity is provided to each of the plurality of containers. An application executing in the container environment uses the identity of the container as opposed to the identity that would otherwise be provided by the operating system. In this manner, the container identity migrates with the container on any platform that hosts the container and associated applications within the container share that identity. This has the further advantage that users of a service or applications within a container are able to interact with the service or application in a consistent manner independently of the physical platform hosting the container.

An example illustrating this further advantage is shown in the attached Exemplary Figure 2. In this example, a business intelligence application is shown that utilizes a web server to provide a user interface. This application may be executing on platform A. The application requires user authentication based on certain keys where the keys may be dependent for example on host name, IP address, MAC address and/or system ID, in other words some identity. The business intelligence application is placed in a container according to the method of new claim 1 and is provided with such a unique identity, in this case a unique identity corresponding to platform A. Then, the business intelligence container is placed on platform B, which is different from platform A and has different identifiers, while the application on platform A is shut down for some reason. Clients of the business intelligence application may then continue to use the same host name or IP address to connect to the web server for user interface and the user authentication continues to function without the need to recreate keys, as the identity of the application is consistent with the original application since as identity of the container the identity of platform A is used. This scenario is applicable to any other platform to which the business intelligence application is moved. Without the use of the invention as defined by claim 1, the business intelligence application would not function on platform B without considerable changes for a number of reasons:

a) The keys used for authentication are dependent on identity parameters defined for platform A;

b) the operating system on platform B may not be configured properly or conformed to the required parameters or revision levels to host the business intelligence application;

c) users of the business intelligence application as a service may require instructions as to how to connect to the service;

d) the business intelligence application is licensed to operate on platform A but not on platform B;

e) the business intelligence application relies on application identity to determine license details.

In summary, the present invention as defined by claim 1 provides a method for executing applications on different platforms having different operating systems and/or different identifiers, whereas document D1 relates to executing applications on a single platform without modifying this single platform.

For the above reasons, taking document D1 into account a person skilled in the art could not arrive at the subject-matter of claim 1 without being inventive. Therefore, the subject-matter of new claim 1 also involves the necessary inventive step.

V.

Regarding claim 15, novelty of claim 15 was already acknowledged in the above-referenced Official Communication. However, in the Official Communication it was stated that the subject-matter of claim 15 is not inventive over prior art document D1.

For similar reasons as already outlined above for claim 1, this opinion cannot be followed, and it is respectfully requested to re-evaluate the question of inventive step of claim 15 taking the following arguments into account.

First of all, it has to be noted that like claim 1 also claim 15 defines a unique identity for each container of files, i.e. each container and the processed within the container share a unique identity as already discussed above. Furthermore, also according to claim 15 the containers comprise associated system files for use in executing the processes in the container, such that in this respect the same argumentation as outlined above for claim 1 applies. Furthermore, the following issues have to be considered:

As already discussed, conventionally processes receive an identity from an operating system, for example by making a corresponding system call. In contrast thereto, according to claim 15 each container of files has its own unique identity associated therewith, which is independent from the underlying operating system, but unique to the container of file. In other words, the applications in the container obtain an identity that is defined by the container and distinct from the operating system and any other application environment which is not taught or hinted at in prior art document D1.

In this respect, as an example it should be noted that as described in the specification when the application in a container environment uses system calls to determine identity parameters, the use of intercepts or "spoofing" may cause alternate values to be returned to the application in a container environment.

Again, the features as defined by claim 15 enable an application to be executed on different platforms which may have different operating systems or may differ in other ways. In particular, as also explained above, the container specific identity obtained by the application in a container moves or migrates with the container. Therefore, when a container is moved from platform A to platform B it can continue to obtain and use the same identity parameters. This allows other applications that are remote to the platform, for example connected to the platform via a network, to interact with the container application in the same way independent of where the container is physically installed. For example, client A locates the container application with IP address X. If the container is moved to a different physical platform, client A continues to locate the container application using IP address X which is one of the possible identifiers given in claim 15.

To explain this in greater detail, reference is made to the example depicted in Exemplary Figure 3 attached to this brief. In this example, an Oracle database is operational on a platform A. This Oracle database application is placed in a container as defined in new claim 15. The container is given an identity that comprises at least one of an Internet Protocol address, a host name, and a MAC address as defined in claim 15. In other words, the container has an identity that corresponds to platform A. As in the previous example, the container is placed on a different platform B, and Oracle on platform A is shut down. The Oracle application in the container environment is able to function without change to the application, the operating system of platform B or the system environment, despite the fact that it has been moved to a different physical platform. Without the use of the invention as defined by claim 15, the Oracle application would not function on platform B without considerable changes for a number of reasons:

a) The Oracle application may be licensed to operate on platform A only.

b) Oracle relies on application identity to determine license detail.

c) The database controls and the database itself is configured to use a specific host name and/or IP address.

d) The operating system on platform B may not be configured properly or conformed to the required parameters and/or revision levels to host an Oracle database.

Therefore, also in this case the invention enables an application to run on different platforms which may differ in identifiers, operating system or other system parameters. On the other hand, as already has been explained above, document D1 relates to running applications on a single system without changing the operating system, in particular the system files associated therewith, on that single system, and not to running systems on different platforms. Therefore, document D1 cannot give a person skilled in the art any hint how to arrive at the subject-matter of new claim 15.

Consequently, also new claim 15 involves the necessary inventive step.

VI.

In summary, it has been shown that the newly filed claims are both novel and inventive over prior art document D1 cited in the above-referenced Official Communication. Furthermore, since a reference to D1 already has been incorporated into the specification, the present application should now be ready for grant. In case, however, the Examining Division is still doubtful regarding the allowability of the present application, it is respectfully requested that a further Communication pursuant to Art. 96(2) EPC be issued or, in case of minor deficiencies, that the undersigned representative be contacted by telephone. As an auxiliary measure, oral proceedings pursuant to Art. 116 EPC are requested should the Examining Division even intend to refuse the present European patent application.

Dipl.-Ing. Hans-Jörg Bänzer
European Patent Attorney

Encl.
New claims 1-17,
New description pages 2, 2a,
Explanatory figures

EPA/EPO/OEB
D-80298 München
+49 89 2399 - 0
FAX +49 89 2399 - 4465

Europäisches Patentamt | European Patent Office | Office européen des brevets

Generaldirektion 2 | Directorate General 2 | Direction Générale 2

Banzer, Hans-Jörg
Kraus & Weisert,
Thomas-Wimmer-Ring 15
80539 München
ALLEMAGNE

Formalities Officer

Name: Lambert

Tel.: 7574

Date
07-08-2006

Reference	Application No./Patent No.
14738EP /nh	04021916.4 - 2211

Applicant/Proprietor
Trigence Corp.

Noting of loss of rights (R. 69(1) EPC)

In the European patent application cited above, the designation(s) of the following Contracting State(s):

AT BE BG CH LI CY CZ DK EE ES FI GR HU IE IT LU MC NL PL PT RO SE SI SK TR

is (are) deemed to be withdrawn because no designation fee in respect of those State(s) was validly paid within the time limits laid down in Article 79(2) and Rules 85a and 25(2) EPC (Art. 79(3) EPC).

Possibility of appeal

If the applicant considers that the finding of the European Patent Office is inaccurate, he may, within **two months** after notification of this communication, apply in writing for a decision on this matter by the European Patent Office (R. 69(2) EPC). The application can only cause the finding to be set aside if loss of rights has not actually occurred.

Examining Division

Europäisches Patentamt • European Patent Office • Office européen des brevets

Registered letter

EPA/EPO/OEB
D-80298 München
+49 89 2399 - 0
FAX +49 89 2399 - 4465

Europäisches Patentamt | **European Patent Office** | **Office européen des brevets**

Generaldirektion 2 | Directorate General 2 | Direction Générale 2

Banzer, Hans-Jörg
Kraus & Weisert,
Thomas-Wimmer-Ring 15
80539 München
ALLEMAGNE

Formalities Officer
Name: Lambert
Tel.: 7574

Date: 24-05-2006

Reference	Application No./Patent No.
14738EP /nh	04021916.4 - 2211

Applicant/Proprietor: Trigence Corp.

Communication pursuant to Rule 85a(1) EPC

The designation fees (for) **AT BE BG CH LI CY CZ DK EE ES FI GR HU IE IT LU MC NL PL PT RO SE SI SK TR** have not been paid in due time (Art. 79(2) EPC).

You can still validly pay the fee(s) within a period of grace of **one month** after notification of this communication, together with a surcharge of 50% (Rule 85a(1) EPC).
The surcharge is limited to a maximum of EUR 680,00 (Art. 2, item 3b Rules relating to Fees.).

If the fee(s) **with surcharge** has (have) not been paid in due time, then, in accordance with Rule 69(1) EPC, you will be informed that

☐ the application is deemed to be withdrawn.

☑ the designation of the above-mentioned Contracting State(s) is deemed to be withdrawn.

The designation fee for each Contracting State designated is **EUR 80,00**.

If the application was filed on or after 1 July 1999, payment of seven times the amount of the designation fee constitutes payment of the designation fees for all contracting states (see OJ EPO 6/1999, 405).

Examination Division

Europäisches Patentamt • European Patent Office • Office européen des brevets

Registered letter

The examination is being carried out on the **following application documents**:

Claims, Numbers

1-17	as originally filed				

Drawings, Sheets

1/17-13/17, 15/17-17/17	as originally filed				
14/17	received on	09.11.2004	with letter of	09.11.2004	

1 Documents

1.1 The following document is referred to in this communication; the numbering will be adhered to in the rest of the procedure:

D1: US 2002/174215 A1 (SCHAEFER STUART) 21 November 2002 (2002-11-21)

2 Summary

2.1 The application does not meet the requirements of Article 84 EPC, because claims 4-17 are **not clear**.

2.2 The lack of clarity notwithstanding, the present application does not meet the requirements of Article 52(1) EPC, because the subject-matter of claims 1-3 is **not new** and thus not allowable in the sense of Article 54(1) and (2) EPC.

2.3 The lack of novelty notwithstanding, the subject-matter of claims 4-17 is **not inventive** (Article 56).

3 Clarity

3.1 The phrase *"...including a kernel a set of associated local system files..."* from claim 1, line 3 should be corrected to read *"...including a kernel, a set of associated local system files..."*.

3.2 The misspelled term *"Mac_address"* used in claims 15 and 16 should be corrected to

"MAC address".

3.3 The term *"MAC address"* used in claims 4, 15 and 16 is **vague and unclear** with regard to Article 84 EPC and should therefore be changed to *"MAC address, a unique identifer of a network node,"*.

3.4 The term *"IP address"* used in claims 4 and 15 should be corrected to *"Internet protocol address"*.

4 Novelty of independent claim 1

4.1 Document **D1** is regarded as closest prior art. It discloses in the original wording of independent claim 1 (reference to the closest prior art is made in parentheses; the original wording of the claim is set in *italic font*):

In a system having a plurality of servers with operating systems that differ, operating in disparate computing environments, wherein each server includes a processor and an operating system including a kernel a set of associated local system files compatible with the processor **(implicit**, since this is a common IT structure**)**, *a method of providing at least some of the servers in the system with secure, executable, applications related to a service, wherein the applications may be executed in a secure environment* **(page 1, paragraph 4, line 3-4**, "...an operating system abstraction and protection layer..."**)**, *wherein the applications each include an object executable by at least some of the different operating systems for performing a task related to the service, the method comprising the steps of:*

storing in memory accessible to at least some of the servers a plurality of secure containers of application software, each container comprising one or more of the executable applications and a set of associated system files required to execute the one or more applications, for use with a local kernel residing permanently on one of the servers **(page 2, paragraph 23, line 7-9**, "...An application bundle is a group of processes..." and **page 2, paragraph 24, line 3-5**, "...serializing its configuration..."**)**;

wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems, the containers of application software excluding a kernel **(figure 1)**, *and*

wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files resident on the

server prior to said storing step **(page 2, paragraph 13, line 7-13**, "...if an application attempts to change the version of a shared object like MSVCRT.DLL this change is localized to the application..."**)**.

4.2 Therefore, the subject-matter of claim 1 is **not new** and thus this claim is not allowable, Articles 52(1) and 54 EPC. Should the applicant be able to identify minor differences or amend the claim by such differences which overcome the above novelty objection, then still the claim can not be considered to be allowable for lack of inventive step, Articles 52(1) and 56 EPC.

5 Inventiveness of independent system claim 15

5.1 Document **D1** is regarded as closest prior art. It discloses in the original wording of independent system claim 15 (reference to the closest prior art is made in parentheses; the original wording of the claim is set in *italic font*; features not explicitly disclosed in the prior art are set ~~strikeout~~)

A computing system for performing a plurality of tasks each comprising a plurality of processes comprising:

a plurality of secure stored containers of associated files accessible to **(implicit**, since each application consists of one or more files**)**, *and for execution on, one or more servers, each container being mutually exclusive of the other, such that read/write files within a container cannot be shared with other containers* **(page 2, paragraph 19, line 1-4**, "...each application is able to run in a private context..." and **page 4 paragraph 40, line 3-4**, "...The present invention provides a virtual file system much like the virtual registry..."; like with the registry, where an application has it's own folders where it can store configuration keys and values, the virtual file system provides the applications own private file system**)**,

each container of files having its own unique identity associated therewith **(page 2, paragraph 22, line 9-11**, "...The core is primarily responsible for managing applications and their context as defined by the configuration files."; the application is identified by it's configuration files**)**,

~~*said identity comprising at least one of an IP address, a host name, and a Mac address;*~~

wherein, the plurality of files within each of the plurality of containers comprise one or more application programs including one or more processes, and associated system files for use in executing the one or more processes, each container having its own execution file associated therewith for starting one or more applications **(implicit**, since it's a well-known design practise to have executable binaries or scripts to launch an application**)**,

in operation, each container utilizing a kernel resident on the server and wherein each container exclusively uses a kernel in an underlying operation system in which it is running and is absent its own kernel **(figure 1)***; and,*

a run time module for monitoring system calls from applications associated with one or more containers and for providing control of the one or more applications **(page 1, paragraph 4, line 9-15**, "...the system continually monitors the use of shared system resources..."**)**.

5.2 The **difference** between **D1** and the subject-matter of claim 1 is that the applications identity comprises one or more network parameters.

5.3 The **objective technical problem** to be solved by the present invention may therefore be regarded as providing an identifier which is unique within a global context.

5.4 The **solution** proposed cannot be considered as involving an inventive step since the incorporation of network parameters like hostname or IP address into an identifier is a common practise used to identify resources (including services) across global networks. The unified resource locator (URL) is widely used to identify processes as for example HTTP servers. Further known standards such as DCOM, CORBA, RPC and various Java technologies use network parameters in the identifiers. Therefore, it would be an obvious alternative to a person skilled in the art to include these parameters into a unique identifier.

5.5 Therefore, the subject-matter of claim 15 is **not inventive** with regard to Article 56 EPC.

6 Novelty of dependent claims

6.1 Dependent claims 2-3 do not appear to contain any additional features which, in combination with the features of any claim to which they refer, meet the requirements of the EPC with respect to **novelty**, the reasons being as follows (reference to the

closest prior art is made in parentheses; the original wording of the claim is set in *italic font*):

6.2 **Claim 2**: an application in a container has no access to system files of applications in other containers **(D1, page 2, paragraph 19)**.

6.3 **Claim 3**: identifying all necessary files required for the application **(D1, page 6, paragraph 59-60**; furthermore, it's obvious for a skilled person that in order to provide an individual execution environment, one of the main tasks is to identify all required components of an application**)** and providing an identifier for the application **(implicit, D1, page 2, paragraph 22)**.

7 Inventiveness of dependent claims

7.1 Dependent claims 4-14 and 16-17 do not appear to contain any additional features which, in combination with the features of any claim to which they refer, meet the requirements of the EPC with respect to **inventive step**, the reasons being as follows (reference to the closest prior art is made in parentheses; the original wording of the claim is set in *italic font*):

7.2 **Claim 4**: as set out previously in item 5.4, the inclusion of the hostname, the IP and MAC address, is an obvious alternative and therefore not inventive.

7.3 **Claim 5 and 7**: distributing various files across different servers within a network is a standard practise known to the person skilled in the art and therefore not inventive.

7.4 **Claim 6 and 17**: monitoring hardware resources is disclosed in **D1 (page 1, paragraph 4, line 13-15)**.

7.5 **Claim 8**: merely discloses an embodiment of an installation procedure without any surprising technical effect and is therefore not inventive.

7.6 **Claim 9-11**: **D1** discloses a graphical user interface to managed the application containers **(page 2, paragraph 22, line 1-6)**.

7.7 **Claim 12-14**: these claims merely specify standard tasks of an installation programm and are therefore not inventive but are rather in juxtaposition to the previous features.

7.8 **Claim 16**: monitoring resource usage and overriding system default settings is disclosed in **D1 (**see **page 1, paragraph 4, line 9-15)**.

8 Conclusion

8.1 It is not at present apparent which part of the application could serve as a basis for a new, allowable claim. Should the applicant nevertheless regard some particular

matter as patentable, an independent claim should be filed taking account of Rule 29(1) EPC.

8.2 In case the applicant files a new set of claims, the applicant is requested to point out and discuss in his letter of reply any difference that would distinguish the subject-matter of the present application from what is disclosed in the available prior art. In particular, the applicant is requested to identify the technical problem that exists in the closest prior art, namely **D1**, describe how the applicant's invention solves this problem, and provide some argument for why this solution would not be obvious to a person skilled in the art.

8.3 When filing amended claims the applicant should at the same time bring the description into conformity with the amended claims. Care should be taken during revision, especially of the introductory portion and any statements of problem or advantage, not to add subject-matter which extends beyond the content of the application as originally filed (Article 123(2) EPC).

8.4 In order to facilitate the examination of the conformity of the amended application with the requirements of Article 123(2) EPC, the applicant is requested to clearly identify the amendments carried out, irrespective of whether they concern amendments by addition, replacement or deletion, and to indicate the passages of the application as filed on which these amendments are based.

8.5 Reference signs in parentheses should be inserted in the claims to increase their intelligibility, Rule 29(7) EPC. This applies to both the preamble and characterising portion (see the Guidelines, C-III, 4.11).

8.6 The applicant is requested to effect the amendments by filing replacement pages for only those pages which have been amended. Unnecessary recasting of the description should be avoided. An amended abstract is not required.

8.7 The applicant should also take account of the requirements of Rule 36(1) EPC. If handwritten amendments are submitted, they should be clearly legible for the printer. In particular, fair copies of the amended pages should be filed in triplicate. Replacement pages containing handwritten amendments should also be filed in triplicate.

8.8 Moreover, it is considered as appropriate in the present case to draft the new independent claim in the two-part form as required by Rule 29(1) EPC, whereby the features known in combination from document **D1** should be placed in the preamble. If the applicant is of the opinion that a two-part form of the claim would be

inappropriate he is invited to provide reasons in his reply. In addition, the applicant should ensure that it is clear from the description which features of the subject-matter of the new independent claim are known from document **D1**; see Guidelines C-III,2.3b.

8.9 To meet the requirements of Rule 27(1)(b) EPC, the document **D1** should be identified in the description and the relevant background art disclosed therein should be briefly discussed.

EPA/EPO/OEB
D-80298 München

+49 89 2399-0
TX 523 656 epmu d
FAX +49 89 2399-4465

Europäisches Patentamt
Generaldirektion 2

European Patent Office
Directorate General 2

Office européen des brevets
Direction Générale 2

Banzer, Hans-Jörg
Kraus & Weisert,
Thomas-Wimmer-Ring 15
80539 München
ALLEMAGNE

Telephone numbers:

Primary Examiner (substantive examination) +49 89 2399-5611

Formalities Officer / Assistant (Formalities and other matters) +49 89 2399-0

Application No.	Ref.	Date
04 021 916.4 - 2211	14738EP /nh	05.05.2006
Applicant: Trigence Corp.		

Communication pursuant to Article 96(2) EPC

The examination of the above-identified application has revealed that it does not meet the requirements of the European Patent Convention for the reasons enclosed herewith. If the deficiencies indicated are not rectified the application may be refused pursuant to Article 97(1) EPC.

You are invited to file your observations and insofar as the deficiencies are such as to be rectifiable, to correct the indicated deficiencies within a period

of 4 months

from the notification of this communication, this period being computed in accordance with Rules 78(2) and 83(2) and (4) EPC.

One set of amendments to the description, claims and drawings is to be filed within the said period on separate sheets (Rule 36(1) EPC).

Failure to comply with this invitation in due time will result in the application being deemed to be withdrawn (Article 96(3) EPC).

Milasinovic, G
Primary Examiner
for the Examining Division

Enclosure(s): 7 page/s reasons (Form 2906)

Registered Letter
EPO Form 2001 07.02CSX

Kraus&Weisert
Patent- und Rechtsanwälte
European Patent and Trademark Attorneys

EPO - Munich
42
10. April 2006

Kraus & Weisert · Thomas-Wimmer-Ring 15 · 80539 München

European Patent Office
Erhardtstraße 27
80469 Munich

Dr. Walter Kraus (– 2002)
Dr.-Ing. Annekäte Weisert (– 2002)
Dr. Thomas Albrecht
Dipl.-Ing. Hans-Jörg Banzer
Dr. Holger Adam
Dr. Inge Hiebl
Dr. Claus Beckmann
Dr. Florian Bertsch
Dr. Andreas Sticht
Dr. Ferdinand Nielsen
Rechtsanwalt

Unser Zeichen / Our Ref.: 14738EP /nh

Bitte in der Antwort angeben
Please refer to in your reply

April 10, 2006

Re: European Patent Application 04 021 916.4-2211
Trigence Corp.

The request for examination (Article 94 EPC) is hereby repeated. It is kindly requested to debit the examination fee as well as the designation fees for the following countries:

Germany
France
United Kingdom

from our deposit account no. 2800.0031 with the European Patent Office.

Dipl.-Ing. Hans-Jörg Banzer
European Patent Attorney

Zur Kasse
1730,- Euro

Encl.
EPO Form 1010

Thomas-Wimmer-Ring 15
80539 München

Tel. +49 (0)89 / 290 60-0
Fax +49 (0)89 / 290 60-111
office@kraus-weisert.de
www.kraus-weisert.de

HypoVereinsbank München
Kto.-Nr.: 667 340 080
BLZ: 700 202 70
IBAN: DE96 7002 0270 0667 3400 80
SWIFT: HYVEDEMM

Postbank München
Kto.-Nr.: 851 02-809
BLZ: 700 100 80
IBAN: DE67 7001 0080 0085 1028 09
SWIFT: PBNKDEFF

P.B.5818 - Patentlaan 2
2280 HV Rijswijk (ZH)
(070) 3 40 20 40
FAX (070) 3 40 30 16

Europäisches Patentamt | **European Patent Office** | **Office européen des brevets**

Generaldirektion 1 | Directorate General 1 | Direction générale 1

Banzer, Hans-Jörg
Kraus & Weisert,
Thomas-Wimmer-Ring 15
80539 München
ALLEMAGNE

EPO Customer Services
Tel.: +31 (0)70 340 45 00

Date
24.10.05

Reference	Application No./Patent No.
14738EP /nh	04021916.4 - 2211

Applicant/Proprietor
Trigence Corp.

Communication pursuant to Rule 50 EPC - reminder of payment of the designation fees (Art. 79(2) EPC) and of the examination fee (Art. 94(2) EPC)

The date on which the European Patent Bulletin mentions the publication of the European search report for the above-mentioned European patent application is: 19.10.05.

Your attention is drawn to Article 79(2) and (3) EPC as well as Article 94(2) and (3) EPC according to which within **SIX MONTHS** after the above-mentioned publication date of the search report

- the designation fee(s) must be paid,
- a written request for examination must be filed as well as the examination fee must be paid. (A written request for examination has been filed already.)

The current rate of the designation fee for each contracting state designated is: EUR 75,00

If the application has been filed on or after 01 July 1999 the payment of seven times the amount of the designation fee is deemed to constitute payment of the designation fees for all contracting states (see OJ EPO 06/1999, 405).

The current rate of the examination fee is: EUR 1430,00

If at least one designation fee and the examination fee are not paid within the period laid down in Article 79(2) or 94(2) EPC, the application shall be deemed to be withdrawn (Arts. 79(3), 94(3) EPC).

Any extension fees are also payable within the above-mentioned period.

Registered letter
EPO Form 1081 01.02

NOTE TO USERS OF THE AUTOMATIC DEBITING PROCEDURE:

1) **Designation fees**
If the application has been filed up to 30 June 1999, the designation fees for the contracting states marked under no. 2 of section 32 of the Request for Grant (EPO Form 1001) will be debited on the last day of the period pursuant to Article 79(2) EPC, unless the EPO receives prior instructions to the contrary.

If the application has been filed on or after 01 July 1999, seven times the amount of the designation fee will be debited on the last day of the period pursuant to Article 79(2) EPC. However, if contracting states are marked under no. 2 of section 32 of the Request for Grant (EPO Form 1001), the designation fees only for these contracting states will be debited unless instructions to the contrary have reached the EPO within the basic period for paying the designation fees.

2) **Examination fee**
Unless the EPO receives prior instructions to the contrary, the examination fee will be debited on the last day of the period for payment.

For further details see the Arrangements for the automatic debiting procedure, Supplement to OJ EPO 02/2002.

RECEIVING SECTION

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 04 02 1916

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

31-08-2005

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2002174215 A1	21-11-2002	AU 2002309834 A2	25-11-2002
		CA 2465880 A1	21-11-2002
		EP 1419437 A1	19-05-2004
		JP 2004533054 T	28-10-2004
		TW 588255 B	21-05-2004
		WO 02093369 A1	21-11-2002
		US 2004268361 A1	30-12-2004

P.B.5818 - Patentlaan 2
2280 HV Rijswijk (ZH)
(070) 3 40 20 40
FAX (070) 3 40 30 16

Europäisches Patentamt | **European Patent Office** | **Office européen des brevets**

Generaldirektion 1 | Directorate General 1 | Direction générale 1

Banzer, Hans-Jörg
Kraus & Weisert,
Thomas-Wimmer-Ring 15
80539 München
ALLEMAGNE

EPO Customer Services
Tel.: +31 (0)70 340 45 00

Date
07.09.05

Reference	Application No./Patent No.
14738EP /nh	04021916.4 - 2211 PCT/

Applicant/Proprietor
Trigence Corp.

COMMUNICATION

The European Patent Office herewith transmits as an enclosure the European search report (under R. 44 or R. 45 EPC) for the above-mentioned European patent application.

If applicable, copies of the documents cited in the European search report are attached.

☑ Additional set(s) of copies of the documents cited in the European search report is (are) enclosed as well.

The following specifications given by the applicant have been approved by the Search Division :

☑ Abstract ☐ Title

☐ The abstract was modified by the Search Division and the definitive text is attached to this communication.

The following figure will be published together with the abstract : 2

Refund of search fee

If applicable under Article 10 Rules relating to fees, a separate communication from the Receiving Section on the refund of the search fee will be sent later.

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 04 02 1916

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
X	US 2002/174215 A1 (SCHAEFER STUART) 21 November 2002 (2002-11-21) * figure 1 * * page 1, paragraph 4 * * page 2, paragraph 13 * * page 2, paragraph 19 * * page 2, paragraph 22 - paragraph 24 * * page 4, paragraph 40 * * page 6, paragraph 59 - paragraph 60 * -----	1-17	G06F9/445

TECHNICAL FIELDS SEARCHED (Int.Cl.7)

G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
Munich	31 August 2005	Milasinovic, G

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

EPA/EPO/OEB
D-80298 München
+49 89 2399-0
TX 523 656 epmu d
FAX +49 89 2399-4465

Europäisches Patentamt
Eingangsstelle

European Patent Office
Receiving Section

Office européen des brevets
Section de Dépôt

Banzer, Hans-Jörg, Dipl.-Ing.
Kraus & Weisert,
Thomas-Wimmer-Ring 15
80539 München
ALLEMAGNE

Datum/Date
02/02/05

Zeichen/Ref./Réf.	Anmeldung Nr./Application No./Demande n°./Patent Nr ./Patent No./Brevet n°.
14738EP /nh	04021916.4-2211 / 1515229

Anmelder/Applicant/Demandeur/Patentinhaber/Proprietor/Titulaire
Trigence Corp.

NOTIFICATION OF EUROPEAN PUBLICATION NUMBER AND INFORMATION ON THE APPLICATION OF ARTICLE 67(3) EPC

The Receiving Section hereby informs you that the technical preparations for publication of the above-mentioned European patent application have been completed.
The provisional protection under Art. 67(1) and (2) EPC in the individual Contracting States becomes effective only when the conditions referred to in Art. 67(3) EPC have been fulfilled (for further information, see EPO brochure "National Law relating to the EPC").

This application will be published on 16.03.05 without the European search report. The publication will be mentioned in European Patent Bulletin number 2005/11
The publication number is: 1515229
The amended title of the invention in the three official languages of the European Patent Office is worded as follows:

Ein System zur Ausführung von Anwendungen innerhalb von Anwendungscontainern
System for executing application sets within application containers
Système pour executer des applications dans des conteneurs d'applications

In all future communications to the EPO, please quote the application number as indicated above, i.e. including the final four figures (which identify the Directorate responsible for the subsequent procedure).
Amendments to a European patent application or European patent must be filed in the language of the proceedings.
REMARK: An issue of the published European patent application will be forwarded to you directly from our printer.

RECEIVING SECTION

Europäisches Patentamt · European Patent Office · Office européen des brevets

Erfindernennung
Designation of inventor
Désignation de l'inventeur

(falls Anmelder nicht oder nicht allein der Erfinder ist) / (where the applicant is not the inventor or is not the sole inventor) / (si le demandeur n'est pas l'inventeur ou l'unique inventeur)

Nr. der Anmeldung oder, falls noch nicht bekannt, Bezeichnung der Erfindung:
Application No. or, if not yet known, title of the invention:
N° de la demande ou, s'il n'est pas encore connu, titre de l'invention :

04 021 916.4

Zeichen des Anmelders oder Vertreters:
Applicant's or representative's reference:
Référence du demandeur ou du mandataire :
(max. 15 Positionen / max. 15 spaces / 15 caractères au maximum)

14738EP /nh

In Sachen der obenbezeichneten europäischen Patentanmeldung nennt (nennen) der (die) Unterzeichnete(n)[1]
In respect of the above European patent application I (we), the undersigned[1]
En ce qui concerne la demande de brevet européen susmentionnée, le(s) soussigné(s)[1]

Trigence Corp.
750 Palladium Drive, Ste 210
Ottawa, Ontario
Canada K2V 1C7

als Erfinder[2]:
do hereby designate as inventor(s)[2]:
désigne(nt) en tant qu'inventeur(s)[2] :

1.) Donn Rochette
3408 30th Avenue
Fenton, Iowa 50539
USA

2.) Paul O'Leary
22 Pentland Crescent
Kanata, Ontario
Canada K2K 1V5

3.) Dean Huffman
13 Pelee Street
Kanata, Ontario
Canada K2M 2R4

☐ Weitere Erfinder sind auf einem gesonderten Blatt angegeben. / Additional inventors indicated on supplementary sheet. / D'autres inventeurs sont mentionnés sur une feuille supplémentaire.

Der (Die) Anmelder hat (haben) das Recht auf das europäische Patent erlangt[3]
The applicant(s) has (have) acquired the right to the European patent[3]
Le(s) demandeur(s) a (ont) acquis le droit au brevet européen[3]

☒ gemäß Vertrag vom / by an agreement dated / par contrat en date du: September 1, 2004

☐ als Arbeitgeber / as employer(s) / en qualité d'employeur(s)

☐ durch Erbfolge / as successor(s) by inheritance / par transfert successoral

Ort/Place/Lieu : Munich

Datum/Date : December 23, 2004

(for Trigence Corp.)

Unterschrift(en) des (der) Anmelder(s) oder Vertreter(s):
Signature(s) of applicant(s) or representative(s):
Signature(s) du (des) demandeur(s) ou du (des) mandataire(s) :

Kraus & Weisert
Patent- und Rechtsanwälte
Dr. T. Albrecht · Dipl.-Ing. H.-J. Banzer · Dr. H. Adam
Dr. I. Hiebl · Dr. C. Beckmann · Dr. F. Nielsen
Thomas-Wimmer-Ring 15 · 80539 München

Dipl.-Ing. Hans-Jörg Banzer

Name des Unterzeichnenden in Druckschrift wiederholen. Bei juristischen Personen bitte die Stellung des (der) Unterzeichneten innerhalb der Gesellschaft in Druckschrift angeben. / Please print name(s) under signature(s). In the case of legal persons, the position of the signatory within the company should also be printed. / Le ou les noms des signataires doivent être indiqués en caractères d'imprimerie. S'il s'agit d'une personne morale, la position occupée au sein de celle-ci par le ou les signataires doit également être indiquée en caractères d'imprimerie.

Fußnoten siehe Rückseite. / Footnotes overleaf. / Le texte des renvois figure au verso.

Kraus & Weisert

Patent- und Rechtsanwälte
European Patent and Trademark Attorneys

Kraus & Weisert · Thomas-Wimmer-Ring 15 · 80539 München

European Patent Office
Erhardtstraße 27
80331 Munich

EPO - Munich
47
23. Dez. 2004

Dr. Walter Kraus (– 2002)
Dr.-Ing. Annekäte Weisert (– 2002)
Dr. Thomas Albrecht
Dipl.-Ing. Hans-Jörg Banzer
Dr. Holger Adam
Dr. Inge Hiebl
Dr. Claus Beckmann
Dr. Ferdinand Nielsen
Rechtsanwalt

Unser Zeichen / Our Ref.: 14738EP /nh

Bitte in der Antwort angeben
Please refer to in your reply

December 23, 2004

Re: European Patent Application 04 021 916.4
Trigence Corp.

We herewith submit the following document(s):

- [x] Designation of Inventor
- [ ] Formal drawings (*** sheets, 3fold)
- [x] Priority documents US 60/502,619 and US 60/512,103
- [ ] Translation of the priority document(s)

Dipl.-Ing. Hans-Jörg Banzer
European Patent Attorney

Thomas-Wimmer-Ring 15
80539 München

Tel. +49 (0)89 / 290 60-0
Fax +49 (0)89 / 290 60-111
office@kraus-weisert.de
www.kraus-weisert.de

HypoVereinsbank München
Kto.-Nr.: 667 340 080
BLZ: 700 202 70
IBAN: DE96 7002 0270 0667 3400 80
SWIFT: HYVEDEMM

Postbank München
Kto.-Nr.: 851 02-809
BLZ: 700 100 80
IBAN: DE67 7001 0080 0085 1028 09
SWIFT: PBNKDEFF

PA 1254931

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE

United States Patent and Trademark Office

December 03, 2004

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THE UNITED STATES PATENT AND TRADEMARK OFFICE OF THOSE PAPERS OF THE BELOW IDENTIFIED PATENT APPLICATION THAT MET THE REQUIREMENTS TO BE GRANTED A FILING DATE UNDER 35 USC 111.

APPLICATION NUMBER: ***60/502,619***
FILING DATE: ***September 15, 2003***

By Authority of the
COMMISSIONER OF PATENTS AND TRADEMARKS

P. SWAIN
Certifying Officer

Please type a plus sign (+) inside this box → [+]

PTO/SB/16 (8-00)
Approved for use through10/31/2002. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PROVISIONAL APPLICATION FOR PATENT COVER SHEET

This is a request for filing a PROVISIONAL APPLICATION FOR PATENT under 37 CFR 1.53(c).

INVENTOR(S)

Given Name (first and middle [if any])	Family Name or Surname	Residence (City and either State or Foreign Country)
Donn	Rochette	Fenton, Iowa, U.S.A.
Dean	Huffman	Kanata, Ontario, Canada
Paul	O'Leary	Kanata, Ontario, Canada

☐ *Additional inventors are being named on th____ separately numbered sheets attached hereto*

TITLE OF THE INVENTION (280 characters max)

DRAG & DROP APPLICATION MANAGEMENT

Direct all correspondence to: **CORRESPONDENCE ADDRESS**

☒ Customer Number 000293 → *Place Customer Number Bar Code Label here*

OR *Type Customer Number here*

☐ Firm or Individual Name					
Address					
Address					
City		State		ZIP	
Country		Telephone		Fax	

ENCLOSED APPLICATION PARTS *(check all that apply)*

☒ Specification *Number of Pages* 22 ☐ CD(s), Number

☒ Drawing(s) *Number of Sheets* 6 ☐ Other (specify)

☐ Application Data Sheet. See 37 CFR 1.76

METHOD OF PAYMENT OF FILING FEES FOR THIS PROVISIONAL APPLICATION FOR PATENT *(check one)*

☒ Applicant claims small entity status. See 37 CFR 1.27.

☒ A check or money order is enclosed to cover the filing fees

☐ The Commissioner is hereby authorized to charge filing fees or credit any overpayment to Deposit Account Number 04-1577

☐ Payment by credit card. Form PTO-2038 is attached.

FILING FEE AMOUNT ($) $80.00

The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government.

☒ No.

☐ Yes, the name of the U.S. Government agency and the Government contract number are: _____

Respectfully submitted,

SIGNATURE

TYPED or PRINTED NAME Ralph A. Dowell

TELEPHONE (703) 415-2555

Date 09/15/03

REGISTRATION NO. *(if appropriate)* 26,868

Docket Number: 14448PRO

USE ONLY FOR FILING A PROVISIONAL APPLICATION FOR PATENT

This collection of information is required by 37 CFR 1.51. The information is used by the public to file (and by the PTO to process) a provisional application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 8 hours to complete, including gathering, preparing, and submitting the complete provisional application to the PTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, Washington, D.C. 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Box Provisional Application, Assistant Commissioner for Patents, Washington, D.C.

P19SMALL/REV05

50357-4

Drag & Drop Application Management

Field of the Invention

The invention relates to computer software. In particular, the invention relates to software that manages the operation of a data center.

Background of the Invention

There has been an unprecedented proliferation of computer systems in the business enterprise. This has been a response to an increase in the number and changing nature of software applications supporting key business processes.

Management of computer systems required to support these applications has become an expensive proposition. This is partially due to the fact that each of the software applications are in most cases hosted on an independent computing platform. The result is an increase in the number of systems to manage.

Summary of the Invention

According to one broad aspect, the invention provides a method of installing a software application on a computing platform. The method includes displaying a software application icon representing the software application and a computing platform icon representing the computing platform. A selection of the software application for installation is received using the software application icon. A selection of the computing platform to which the software application is to be installed is received using the computing platform icon. Installation of the selected software application is then initiated on the selected computing platform.

In some embodiments, the computing platform is a remote computing platform.

In some embodiments, the selection of the software application is received with the use of a graphical user interface in response to the software application icon being pointed to using a pointing device.

In some embodiments, the pointing device is a mouse.

In some embodiments, the selection of the computing platform is received in response to the software application icon being dragged over the computing platform icon and the software application icon being dropped.

In some embodiments, the installation of the selected software application on the selected computing platform is initiated in response to the software application icon being dropped over the computing platform icon.

In some embodiments, upon initialization, the selected computing platform is tested to determine whether it is a valid computing platform.

In some embodiments, upon initialization, a user account is created for the selected software application.

In some embodiments, upon initialization, files specific to the selected application software are installed on the selected computing platform.

In some embodiments, upon initialization, file access permissions are set to allow a user to access the application.

In some embodiments, upon initialization, a console on the selected computing platform is updated with information indicating that the selected software application is resident on the selected computing platform.

In accordance with a second broad aspect, the invention provides a method of de-installing a software application from a computing platform. The method includes displaying a software application icon representing the software application and a computing platform icon representing the computing platform on which the software application is installed. A selection of the software application for de-installation using the software application icon is received. A selection of the computing platform from which the software application is to be de-installed using the computing platform icon is also received. De-installation of the selected software application from the selected computing platform is then initiated.

In some embodiments, the computing platform is a remote computing platform.

In some embodiments, the displaying comprises displaying the software application as being installed on the computing platform with the use of the software application icon and the computing platform icon.

In some embodiments, the selection of the software application is received with the use of a graphical user interface in response to the software application icon being pointed to using a pointing device.

In some embodiments, the pointing device is a mouse.

In some embodiments, the selection of the computing platform is in response to the software application icon being dragged away from the computing platform icon and the software application icon being dropped.

In some embodiments, upon initialization, the selected computing platform is tested to determined whether it is a valid computing platform for de-installation of the software application.

In some embodiments, upon initialization, any process associated with the selected software application is stopped.

In some embodiments, upon initialization, data file changes specific to the software application are copied back to a storage medium from which the data file changes originated prior to installation.

In some embodiments, upon initialization, a user account associated with the selected software application is removed.

In some embodiments, upon initialization, a console on the computing platform is updated with information indicating that the selected software application is no longer resident on the selected computing platform.

In accordance with a third broad aspect, the invention provides an article of manufacture comprising a computer usable medium having computer readable program code means embodied therein for installing a software application on a computing platform. The computer readable code means in the article of manufacture has computer readable code means for performing any of the above method steps for installation.

In accordance with a fourth broad aspect, the invention provides an article of manufacture comprising a computer usable medium having computer readable program code means embodied therein for de-installing a software application from a computing platform. The computer readable code means in the article of manufacture has computer readable code means for performing any of the above method steps for de-installation.

In accordance with a fifth broad aspect, the invention provides a GUI (Graphical User Interface) adapted to perform any of the above method steps for installation or de-installation.

In accordance with a sixth broad aspect, the invention provides a GUI adapted to display a software application icon representative of a software application available for installation and display a computing platform icon representative of a computing platform. The GUI is responsive to a selection of the software application icon and the computing platform icon by initiating installation of the software application on the computing platform.

In accordance with a seventh broad aspect, the invention provides a GUI adapted to display a software application icon representative of a software application and display a computing platform icon representative of a computing platform, the GUI being responsive to a selection of the software application icon and the computing platform icon by initiating de-installation of the software application from the computing platform.

In accordance with an eighth broad aspect, the invention provides a GUI adapted to display a software application icon representative of a software application

installed on a computing platform, the GUI being responsive to a selection of the software application icon by initiating de-installation of the software application from the computing platform.

In some embodiments of the invention, the selection is made using a drag and drop operation.

In accordance with an ninth broad aspect, the invention provides a method of de-installing a software application from a computing platform. The method includes displaying a software application icon representing the software application installed on a computing platform. A selection of the software application for de-installation from the computing platform is received using the software application icon. The de-installation of the selected software application from the computing platform is then initiated.

In some embodiments, the selection of the application icon is in response to the software application icon being dragged away.

In some embodiments, the de-installation of the selected software application from the computing platform is initiated in response to the software application icon being dropped.

Accordingly, the invention relates, in part to methods of installing and removing software applications through a selection such as, for example, a graphical drag and drop operation.

In some embodiments of the invention, functions of the GUI support the ability to select an object, move it and initiate an operation when the object is placed on a specific

destination. This process has supported operations including the copy and transfer of files.

Some embodiments of the invention extend this operation to allow software applications, represented by a graphical icon, to be installed in working order following a drag and drop in a graphical user interface.

Brief Description of the Drawings

Preferred embodiments of the invention will now be described with reference to the attached drawings in which:

Figure 1 is a schematic showing the operation of the invention, according to an embodiment of the invention;

Figure 2 is a schematic showing software application icons collected in a centralized file storage medium and a remote computing platform icon, according to another embodiment of the invention;

Figure 3 is a screen snapshot of an implementation according to the schematic of Figure 2;

Figure 4 is a flow chart of a method of initializing icons of Figure 3;

Figure 5 is a flow chart of a method of installing a software application on a computing platform in response to a drag & drop operation of Figure 2; and

Figure 6 is a flow chart of a method of de-installing a software application from a computing platform in response to a drag & drop operation of Figure 2.

Detailed Description of the Preferred Embodiments

The following definitions are used herein:

Storage repository: A centralized disk based storage containing application specific files that make up a software application.

Computing platform: A computer hardware platform having an operating system is referred to as a computing platform. As such, a computing platform is suitably configured and ready to host software applications.

GUI (Graphical User Interface): A computer-based display that presents a graphical interface by which a user interacts with a computing platform by means of icons, windows, menus and a pointing device is referred to as a GUI.

Icon: An icon is an object supported by a GUI. Normally an icon associates an image with an object that can be operated on through a GUI.

Aspects of the invention include:

- GUI supporting drag & drop operations
- Icons
- Storage medium
- Console
- Network connections
- One or more computing platforms

While software applications are represented as graphical icons, they are physically contained in a storage medium. Such a storage medium consists, for example, of disk-

based file storage on a server accessible through a network connection.

A computing platform is a computer with an installed operating system capable of hosting software applications.

When a software application icon is "dropped" on a computing platform the applications associated with the icon are installed in working order on the selected platform. The computing platform is generally remote with respect to either the platform hosting the graphical interface or the server hosting the storage medium. This topology is not strictly required, but rather a likely scenario.

Referring to Figure 1, shown is a schematic showing the operation of the invention, according to an embodiment of the invention. A graphical icon representing a specific software application is displayed through a graphical user interface. Also displayed is an icon representing a remote computing platform upon which a software application can be hosted. Only one computing platform icon is shown; however, it is to be understood that several computing platform icons each representing a respective remote or local computing platform may also be displayed. Similarly, several software application icons may be displayed depending on the number of software applications available. The software application is selected, through the use of a graphical user interface, by pointing to its software application icon and using a mouse click (or other pointing device). The software application icon is dragged over top of the remote computing platform icon and dropped. The drop initiates the operation of installing software applications on the remote computing platform.

Referring to Figure 2, shown is a schematic showing software applications icons collected in a centralized file

storage medium and a remote computing platform icon, according to another embodiment of the invention. The software applications icons collected in the file storage medium, as shown, are graphical icons each representing respective software applications which are entries in the file storage medium. The computing platform icon represents a computing platform available to host any one or more of the software applications represented by the software icons. When one of the software application icons is selected, dragged, and dropped on a computing platform icon the respective software applications installed on the remote computing platform corresponding to the computing platform icon.

Referring to Figure 3, shown is a screen snapshot of an implementation according to the schematic of Figure 2. The screen snap shot shows, as an example implementation, software application and computing platform icons. Available software applications corresponding to software application icons fs0, fs0, and fs1 are grouped under the heading "Data Center 1". Computing platforms available to host software applications are featured under the heading "Repository" and are represented by computing platform icons CRM, MySQL, WebServer, Video, and Skeleton. Operations involve selecting a software application icon, dragging it over a candidate computing platform icon and releasing, or dropping it on the computing platform icon. The selected software application will then be installed in working order on the selected computing platform. The reverse is true for removal of a software application from a selected computing platform. De-installation is accomplished by selecting an application installed on a compute platform and dragging to the repository. The application in question is shown to be associated with a compute platform in graphical fashion. In one embodiment, as shown in figure 3, applications are associated with a compute platform in hierarchical order, or

in a tree view. An application connected to a compute platform as root in a tree view is selected and dropped onto the repository. This initiates the de-install operation.

Referring to Figure 4, shown is a flow chart of a method of initializing the icons of Figure 3. An icon such as a computing platform icon or software application icon is created as a GUI (Graphical User Interface) object. Drop handler operations are then associated to the computing platform icon. In particular, any operation required for installation is associated with the icon.

1.Installation

As discussed above with reference to Figures 1 to 3, the installation process is initiated by a drag & drop of a software application icon, from a storage medium, to a top of a computing platform icon. An install drop handler implements the installation of the software application.

The install drop handler performs several tasks on the destination computing platform:

- Tests whether the computing platform is a valid computing platform;
- Creates a user account for the software application;
- Installs application specific files so that they are accessible to the remote computing platform;
- Sets file access permissions such that a new user can access its applications;
- Starts a first application; and

- Updates a console showing that the selected software application is resident on the selected computing platform.

These steps will now be described with reference to Figure 5 which is a flow chart of a method of installing a software application on a computing platform in response to the drag & drop operation of Figure 2. As discussed above, in operation the installation process is invoked when a software application icon is dropped on a computing platform icon. The method steps of Figure 5 are performed by an install drop handler. During installation, a verification is performed to determine whether the selected computing platform is a valid candidate for installing a selected software application. If the computing platform is a valid candidate, a user account is created for the software application; otherwise, the installation process ends. Once the user account is created, application files associated with the software applications are installed on the selected computing platform so that they are accessible to the computing platform. File access permissions are then set such that one or more new users can access the application being installed. A first application is then started. In some embodiments, an application, for example accounting or payroll, represent several programs/processes. In order to start the accounting/payroll service a first application is started that MAY in turn start other programs/processes. The first program is started. It is then up to the specific service, accounting/payroll, to start any other services it requires.

A console on the selected computing platform on which the software application is being installed is then updated with information to show that the selected software application is resident on the selected computing platform. The console is

operational on any desktop computing platform for example, Unix, Linux and Windows.

2.Removal of application software from a computing platform

With reference to Figure 3, the removal process is initiated by a drag & drop operation of a software application icon from a computing platform icon to the repository. For this purpose each computing platform icon shows, with the use of associated software application icons (not shown in Figure 3), each application software installed on the computing platform.

The de-installation of application software from a selected computing platform is done by a remove drop handler which performs the following tasks on the selected computing platform:

- Tests whether the computing platform is a valid computing platform; (Tests are made to verify whether the computing platform is operational. For example, it may have been taken off-line due to a problem or routine maintenance. If so, then applications should not be removed or installed until this state changes.)
- Stops processes associated with the software application;
- Copies application specific data file changes from the computing platform back to the storage medium;
- Removes user accounts associated with the software application; and
- Updates the console to show that the selected software application is resident in the repository.

These steps will now be described with reference to Figure 6 which is a flow chart of a method of de-installing a software application from a computing platform in response to a drag & drop operation of Figure 2. The state of the platform is verified to determine whether it is valid. The state includes, for example, on-line, off-line (a problem state) or maintenance (off-line, but for a scheduled task). If the state of the platform is valid and a process or processes associated with a software application selected to be de-installed are stopped; otherwise the de-installation process ends. Once the processes are stopped, application specific data file changes are copied from the computing platform back to the storage medium. This is a process of capturing changes that may have taken place while the application ran and that need to be saved for future instances when the application runs again. For example, payroll may be run every other week. Changes made to the system, accrued amounts, year to date payments, etc. will need to be saved so that when payroll is run the next time these values are updated. It may be required (depending on how the operator configures a system) to copy changes made when payroll ran back to the repository so that the next time payroll is run these values are installed along with the application.

Any user account associated with the selected software application is removed and a console on the computing platform from which the selected software application is being de-installed is updated with information to show that the selected software application is resident in the repository.

Using the method steps of Figures 5 and 6, software applications are therefore installed on a computing platform and removed from the computing platform through a graphical user interface drag and drop operation.

A number of programming languages may be used to implement programs used in embodiments of the invention. Some example programming languages used in embodiments of the invention include, but are not limited to, C++, Java and a number of scripting languages including TCL/TK and PERL.

Installation of software applications is preferably performed on computing platforms that are remote from a console computing platform. However, some embodiments of the invention also have installation of software applications performed on a computing platform that is local to a console computing platform.

Numerous modifications and variations of the present invention are possible in light of the above teachings. It is therefore to be understood that within the scope of the appended claims, the invention may be practised otherwise than as specifically described herein.

50357-4

WE CLAIM:

1. A method of installing a software application on a computing platform, the method comprising:

displaying a software application icon representing the software application and a computing platform icon representing the computing platform;

receiving a selection of the software application for installation using the software application icon;

receiving a selection of the computing platform to which the software application is to be installed using the computing platform icon; and

initiating the installation of the selected software application on the selected computing platform.

2. A method according to claim 1 wherein the computing platform is a remote computing platform.

3. A method according to claim 1 or 2 wherein the selection of the software application is received with the use of a graphical user interface in response to the software application icon being pointed to using a pointing device.

4. A method according to claim 3 wherein the pointing device is a mouse.

5. A method according to claim 3 or 4 wherein the receiving a selection of the computing platform is in response to the software application icon being dragged over the computing platform icon and the software application icon being dropped.

6. A method according to claim 5 wherein the initiating the installation of the selected software application on the selected computing platform is in response to the software application icon being dropped over the computing platform icon.

7. A method according to anyone of claims 1 to 6 further comprising, upon initialization, testing whether the selected computing platform is a valid computing platform.

8. A method according to anyone of claims 1 to 7 further comprising, upon initialization, creating a user account for the selected software application.

9. A method according to anyone of claims 1 to 8 further comprising, upon initialization, installing files specific to the selected application software on the selected computing platform.

10. A method according to claim 9 further comprising, upon initialization, setting file access permissions to allow a user to access the application.

11. A method according to anyone of claims 1 to 10 further comprising starting the selected software application.

12. A method according to anyone of claims 1 to 11 further comprising, upon initialization, updating a console on the selected computing platform with information indicating that the selected software application is resident on the selected computing platform.

13. A method of de-installing a software application from a computing platform, the method comprising:

displaying a software application icon representing the software application and a computing platform icon representing the computing platform on which the software application is installed;

receiving a selection of the software application for de-installation using the software application icon;

receiving a selection of the computing platform from which the software application is to be de-installed using the computing platform icon; and

initiating the de-installation of the selected software application from the selected computing platform.

14. A method according to claim 13 wherein the computing platform is a remote computing platform.

15. A method according to claim 13 or 14 wherein the displaying comprises displaying the software application as being installed on the computing platform with the use of the software application icon and the computing platform icon.

16. A method according to anyone of claims 13 to 15 wherein the selection of the software application is received with the use of a graphical user interface in response to the software application icon being pointed to using a pointing device.

17. A method according to claim 16 wherein the pointing device is a mouse.

18. A method according to claim 16 or 17 wherein the displaying a software application icon comprises displaying

software application icon as being installed on the computing platform and wherein the selection of the application icon and the selection of the computing platform are in response to the software application icon being dragged away from the software application icon.

19. A method according to claim 18 wherein the initiating the de-installation of the selected software application from the selected computing platform is in response to the software application icon being dropped.

20. A method according to anyone of claims 13 to 19 further comprising, upon initialization, testing whether the selected computing platform is a valid computing platform for de-installation of the software application.

21. A method according to anyone of claims 13 to 20 further comprising, upon initialization, stopping any process associated with the selected software application.

22. A method according to anyone of claims 13 to 21 further comprising, upon initialization, copying data file changes specific to the software application back to a storage medium from which the data file changes originated prior to installation.

23. A method according to anyone of claims 13 to 22 further comprising, upon initialization, removing a user account associated with the selected software application.

24. A method according to anyone of claims 13 to 23 further comprising, upon initialization, updating a console on the computing platform with information indicating that the

selected software application is no longer resident on the selected computing platform.

25. An article of manufacture comprising:

a computer usable medium having computer readable program code means embodied therein for installing a software application on a computing platform, the computer readable code means in said article of manufacture comprising:

computer readable code means for performing the method steps of any one of claims 1 to 12.

26. An article of manufacture comprising:

a computer usable medium having computer readable program code means embodied therein for de-installing a software application from a computing platform, the computer readable code means in said article of manufacture comprising:

computer readable code means for performing the method steps of any one of claims 13 to 24 and 32 to 34.

27. A GUI (Graphical User Interface) adapted to perform the method steps of anyone of claims 1 to 24 and 32 to 34.

28. A GUI (Graphical User Interface) adapted to display a software application icon representative of a software application available for installation and display a computing platform icon representative of a computing platform, the GUI being responsive to a selection of the software application icon and the computing platform icon by initiating installation of the software application on the computing platform.

29. A GUI (Graphical User Interface) adapted to display a software application icon representative of a software

application and display a computing platform icon representative of a computing platform, the GUI being responsive to a selection of the software application icon and the computing platform icon by initiating de-installation of the software application from the computing platform.

30. A GUI (Graphical User Interface) adapted to display a software application icon representative of a software application installed on a computing platform, the GUI being responsive to a selection of the software application icon by initiating de-installation of the software application from the computing platform.

31. A GUI (Graphical User Interface) according to anyone of claims 29 to 30 wherein the selection is in response to a drag and drop operation.

32. A method of de-installing a software application from a computing platform, the method comprising:

displaying a software application icon representing the software application installed on a computing platform;

receiving a selection of the software application for de-installation from the computing platform using the software application icon;

initiating the de-installation of the selected software application from the computing platform.

33. A method according to claim 32 wherein the selection of the application icon is in response to the software application icon being dragged away.

34. A method according to claim 33 wherein the initiating the de-installation of the selected software application from the computing platform is in response to the software application icon being dropped.

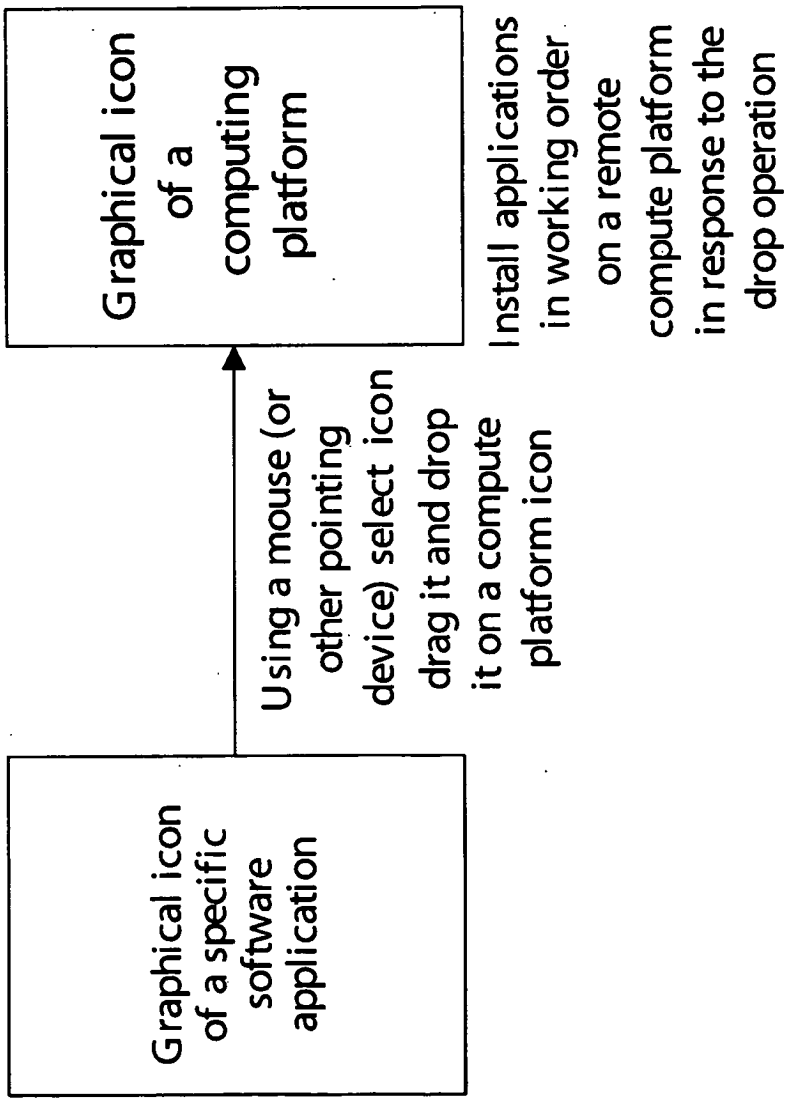


FIG. 1

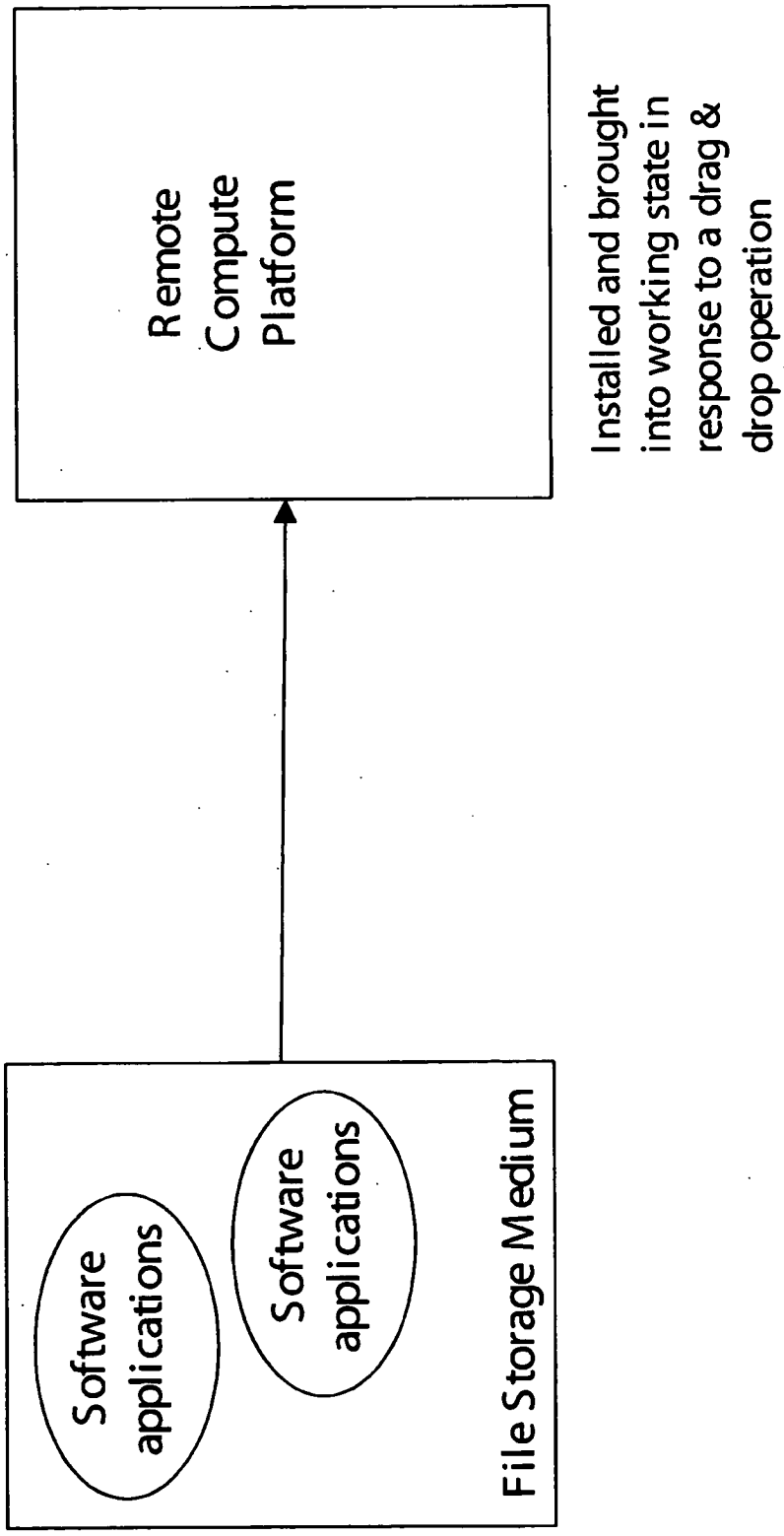
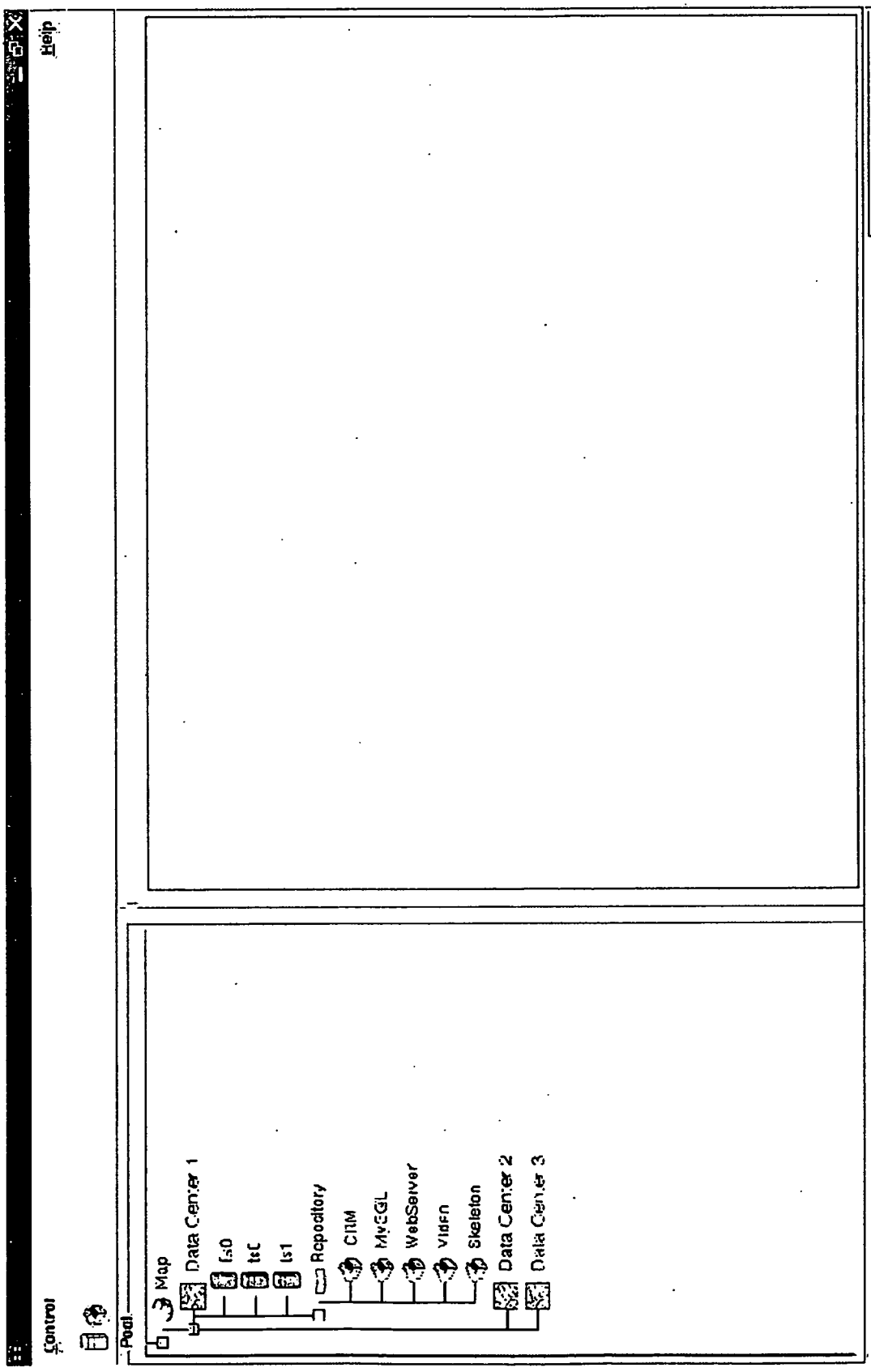


FIG. 2

FIG. 3



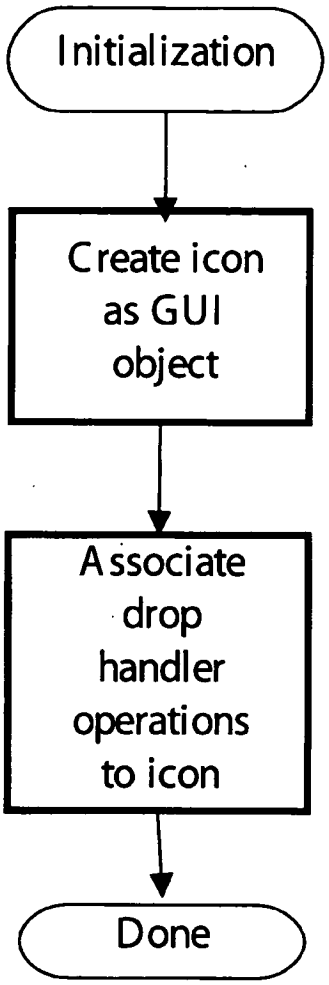


FIG. 4

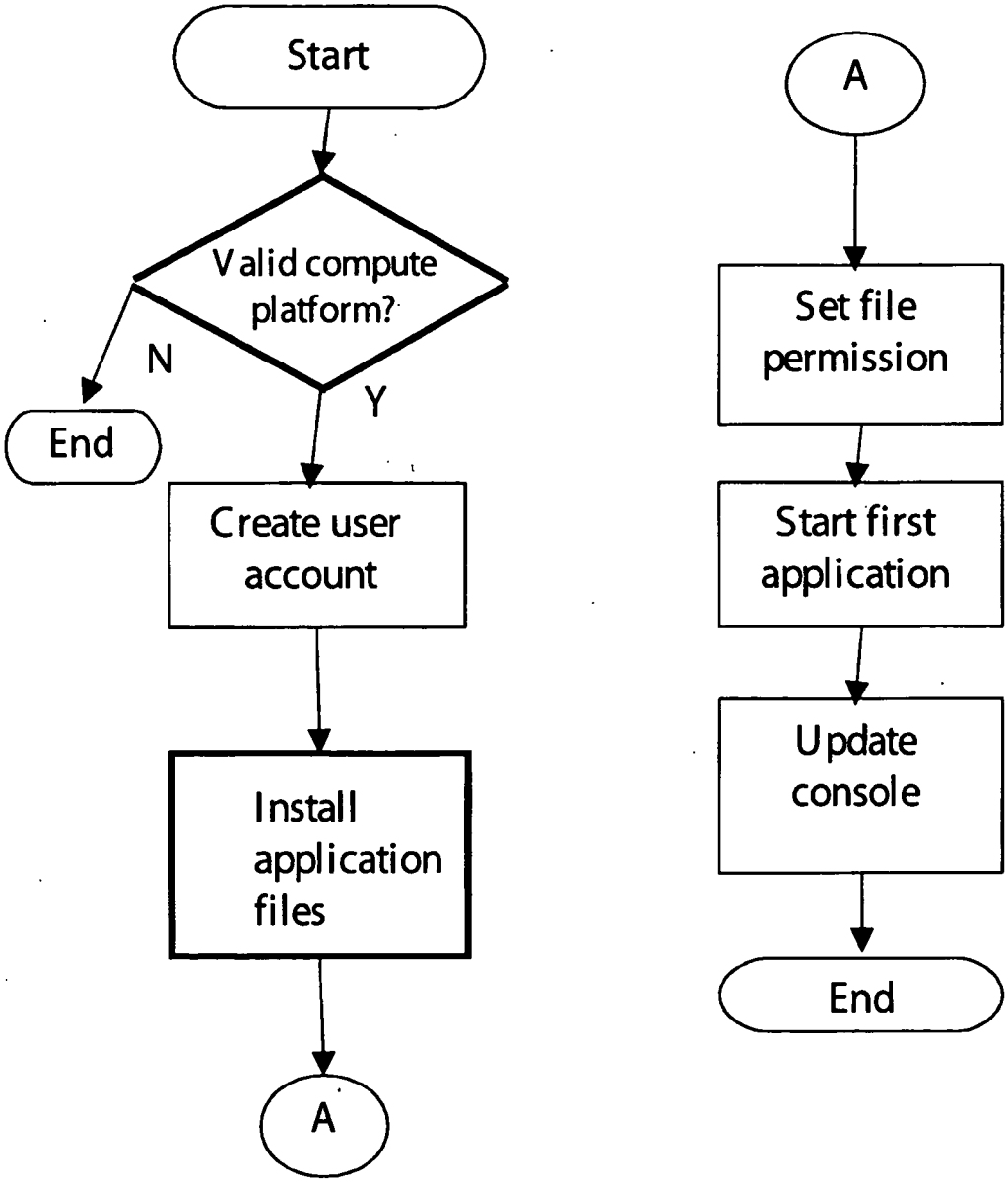


FIG. 5

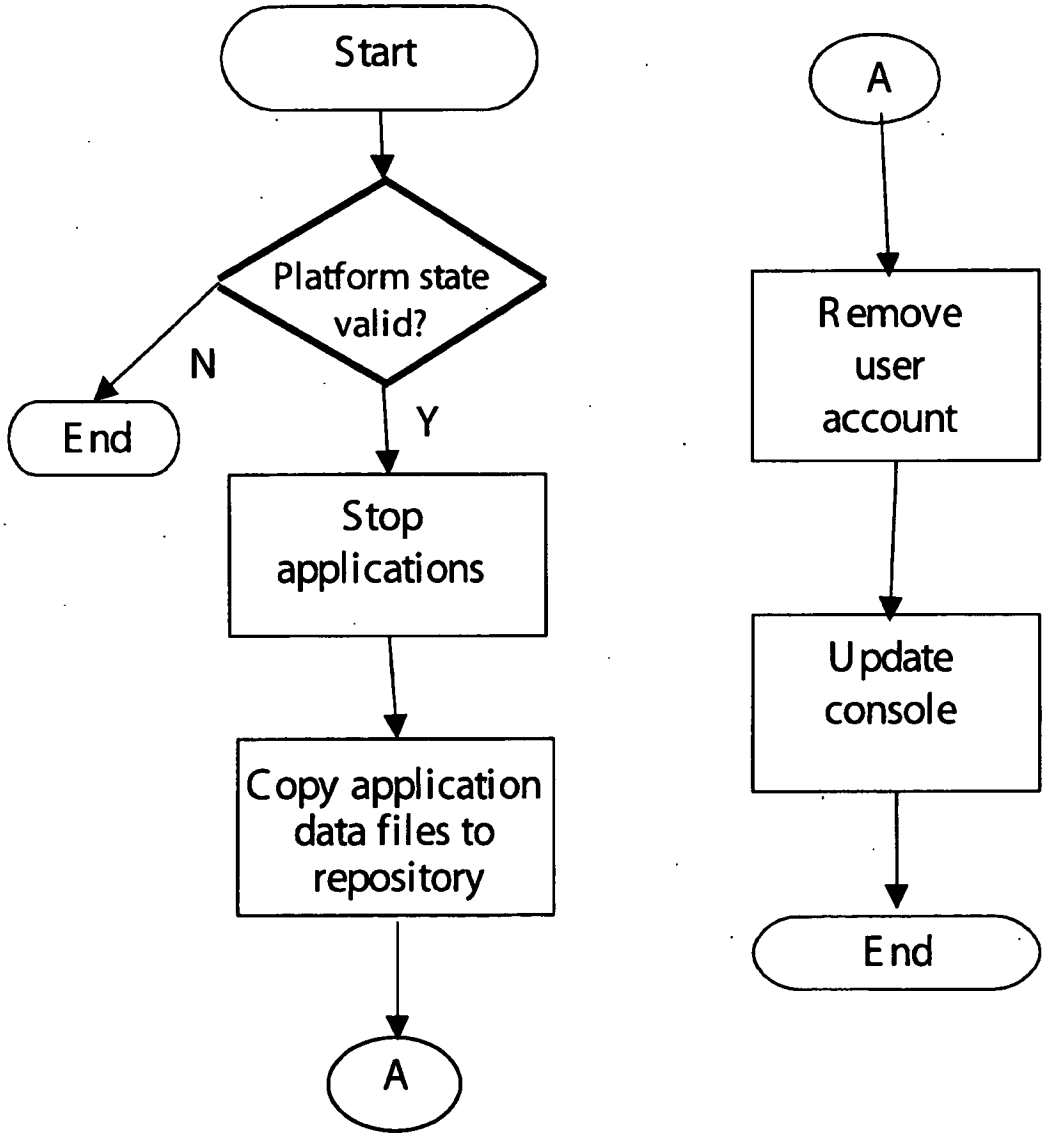


FIG. 6

PA 1254808

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE

United States Patent and Trademark Office

December 06, 2004

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THE UNITED STATES PATENT AND TRADEMARK OFFICE OF THOSE PAPERS OF THE BELOW IDENTIFIED PATENT APPLICATION THAT MET THE REQUIREMENTS TO BE GRANTED A FILING DATE UNDER 35 USC 111.

APPLICATION NUMBER: ***60/512,103***
FILING DATE: ***October 20, 2003***

By Authority of the
COMMISSIONER OF PATENTS AND TRADEMARKS

H. L. Jackson

H. L. JACKSON
Certifying Officer

Please type a plus sign (+) inside this box → [+]

PTO/SB/16 (5-03)
Approved for use through 4/30/2003. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PROVISIONAL APPLICATION FOR PATENT COVER SHEET

This is a request for filing a PROVISIONAL APPLICATION FOR PATENT under 37 CFR 1.53(c).

INVENTOR(S)

Given Name (first and middle [if any])	Family Name or Surname	Residence (City and either State or Foreign Country)
Donn	Rochette	Fenton, Iowa, U.S.A.
Dean	Huffman	Kanata, Ontario, Canada
Paul	O'Leary	Kanata, Ontario, Canada

☐ *Additional inventors are being named on th____ separately numbered sheets attached hereto*

TITLE OF THE INVENTION (280 characters max)

SYSTEM FOR CONTAINERIZATION OF APPLICATION SETS

CORRESPONDENCE ADDRESS

Direct all correspondence to:

☒ Customer Number 000293 →

OR *Type Customer Number here*

Place Customer Number Bar Code Label here

☐ Firm *or* Individual Name					
Address					
Address					
City		State		ZIP	
Country		Telephone		Fax	

ENCLOSED APPLICATION PARTS *(check all that apply)*

☒ Specification *Number of Pages* 27

☒ Drawing(s) *Number of Sheets* 3

☐ Application Data Sheet. See 37 CFR 1.76

☐ CD(s), Number

☐ Other (specify)

METHOD OF PAYMENT OF FILING FEES FOR THIS PROVISIONAL APPLICATION FOR PATENT *(check one)*

☒ Applicant claims small entity status. See 37 CFR 1.27.

☒ A check or money order is enclosed to cover the filing fees

☐ The Director is hereby authorized to charge filing fees or credit any overpayment to Deposit Account Number

☐ Payment by credit card. Form PTO-2038 is attached.

FILING FEE AMOUNT ($): **$80.00**

The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government.

☒ No.

☐ Yes, the name of the U.S. Government agency and the Government contract number are: _____

Respectfully submitted,

SIGNATURE

Date 10/20/03

REGISTRATION NO. *(if appropriate)* 26,868

Docket Number: 14487PRo

TYPED or PRINTED NAME Ralph A. Dowell

TELEPHONE (703) 415-2555

USE ONLY FOR FILING A PROVISIONAL APPLICATION FOR PATENT

This collection of information is required by 37 CFR 1.51. The information is used by the public to file (and by the PTO to process) a provisional application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 8 hours to complete, including gathering, preparing, and submitting the complete provisional application to the PTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Mail Stop Provisional Application, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need asssitance in completing the form, call 1-800-PTO-9199 and select option 2.

P19SMALL/REV05

50357-5

SYSTEM FOR CONTAINERIZATION OF APPLICATION SETS

Field of the Invention

The invention relates to computer software. In particular, the invention relates to management and deployment of server applications.

Background of the Invention

Computer systems are designed in such a way that application programs share common resources. It is traditionally the task of an operating system to provide a mechanism to safely and effectively control access to shared resources. This is the foundation of multi-tasking systems that allow multiple disparate applications to co-exist on a single computer system.

The current state of the art creates a situation where a collection of applications each designed for a distinct function must be separated with each application installed on an individual computer system. In some cases this is driven by conflict over shared resources, such as network port numbers. In other situations the separation is driven by the need to securely separate data (files contained on disk-based storage) and/or applications between disparate users. In yet other situations, the separation is driven by the reality that certain applications require a specific version of operating system facilities and as such will not co-exist with applications that require another version.

As a computer system architecture is applied to support specific services it inevitably requires that separate systems be deployed for each application set. This fact coupled with increased demand for support of additional application sets results in a significant increase in the

number of computer systems being deployed. Such deployment makes it quite costly to manage the number of systems required to support several applications.

Summary of the Invention

In accordance with a first broad aspect, the invention provides a method of establishing a secure environment for executing, on a computer system, a plurality of applications that require shared resources. The method involves, for each group of a plurality of groups, associating at least one respective application of the plurality of applications with the group. A respective resource allocation is associated with the group for the at least one respective application associated with the group to allow the at least one respective application associated with the group to be executed on the computer system according to the respective resource allocation without conflict with the at least one respective application associated with other groups of the plurality of groups.

In some embodiments of the invention, a group of applications having one or more applications resides in a container. A container, and therefore the applications within the container, are provided with a secure environment from which to execute.

In some embodiments of the invention, each group of applications is provided with a secure storage medium.

In some embodiments of the invention, for each group of the plurality of groups, the method involves executing on the computer system the at least one respective applications associated with the group according to the respective resource allocation associated with the group.

In some embodiments of the invention, for each group of the plurality of groups, the method involves containerizing the at least one respective application associated with the group into a respective secure application container for the group, the respective secure application container being storable in a storage medium.

In some embodiments of the invention, for each group of the plurality of groups, application files for the at least one respective application associated with the group are containerized into the respective secure application container for the group.

In some embodiments of the invention, for each group of the plurality of groups, application system calls for the at least one respective application associated with the group are containerized into the respective container for the group.

In some embodiments of the invention, for each group of the plurality of groups, the at least one respective application of the plurality of applications are associated with the group according to classes of service.

In some embodiments of the invention, for each group of the plurality of groups, the at least one respective application of the plurality of applications are associated with the group according to classes of service comprising specific application services.

In some embodiments of the invention, the method involves exporting one or more of the respective secure application containers to a remote computer system.

In some embodiments of the invention, each respective secure application container has data files for the at least one application within the respective secure application

container. The method involves making the data files within one of the respective secure application containers inaccessible to the at least one respective application within another one of the respective secure application containers.

In some embodiments of the invention, for each respective secure application container, the method involves providing the at least one respective application within the respective secure application container with a respective root file system.

In some embodiments of the invention, for each group of the plurality of groups, the method involves associating a respective IP address for the group.

In some embodiments of the invention, for each group of the plurality of groups, the method involves associating resource limits for the at least one respective application associated with the group.

In some embodiments of the invention, for each group of the plurality of groups, the method involves associating resource limits comprising any one or more of limits on memory, CPU bandwidth, Disk size and bandwidth and Network bandwidth for the at least one respective application associated with the group.

In some embodiments of the invention, each group of the plurality of groups forms a secure application container having files related to the at least one respective application associated with the group. For each secure application container, the method involves determining whether resources available on the computer system can accommodate the respective resource allocation associated with the group comprising the secure application container. If the computer system can

accommodate the respective resource allocation associated with the group forming the secure application container, the secure application container is exported to the computer system.

In some embodiments of the invention, if one or more secure application containers of the secure application containers are already installed on the computer system, the method involves determining whether the computer system has enough resources available to accommodate the one or more secure application containers which are already installed and the secure application container being exported. If there are enough resources available, the resources are distributed between the one or more secure application containers and the secure application container being exported to provide resource control.

In some embodiments of the invention, in determining whether resources available on the computer system can accommodate the respective resource allocation, the method involves verifying whether the computer system supports any specific hardware required by the secure application container to be exported.

In some embodiments of the invention, for each group of the plurality of groups, in associating a respective resource allocation with the group, the method involves associating resource limits for the at least one respective application associated with the group. The method also involves during execution on the computer system: monitoring resource usage of the at least one respective application associated with the group; intercepting system calls, made by the at least one respective application associated with the group, from user mode to kernel mode; comparing the monitored resource usage of the at least one respective application

associated with the group with the resource limits; and forwarding the system calls to a kernel on the basis of the comparison between the monitored resource usage and the resource limits.

In some embodiments of the invention, the method involves searching for application specific files associated with applications that are installed and working on computer systems.

In some embodiments of the invention, the method involves extracting the application specific files from the computer systems to create application sets ready to install in secure application containers on the computer system.

In some embodiments of the invention, the method involves converting the application specific files into an intermediate format that allows the applications that are installed and working on computer systems to be installed in secure application containers on the computer system.

In some embodiments of the invention, the method involves copying the application specific files, and related data and configuration information to a file storage medium.

In some embodiments of the invention, the copying is achieved with the use of a user interface in which application programs are displayed and selected for copying to a storage medium.

In some embodiments of the invention, the application specific files, and related data and configuration information are made available for installation into secure application containers on a remote computer system.

In some embodiments of the invention, the method involves installing at least one of the plurality of applications in a root file system.

In accordance with a second broad aspect, the invention provides a method of establishing a secure environment for executing, on a computer system, a plurality of applications that require shared resources. The method involves, for each secure application container of a plurality of secure application containers, containerizing at least one respective application of the plurality of applications into the secure application container. A respective resource allocation is associated with the secure application container for the at least one respective application containerized within the secure application container to allow the at least one respective application containerized within the secure application container to be executed on the computer system according to the respective resource allocation without conflict with the at least one respective application containerized within other secure application containers of the plurality of secure application containers.

In accordance with a third broad aspect, the invention provides a computer usable medium having computer readable program code means embodied therein for establishing a secure environment for executing, on a computer system, a plurality of applications that require shared resources. The computer readable code means in the article of manufacture has, for each group of a plurality of groups, computer readable code means for associating at least one respective application of the plurality of applications with the group; and computer readable code means for associating a respective resource allocation with the group for the at least one respective application associated with the group to allow the at least one

respective application associated with the group to be executed on the computer system according to the respective resource allocation without conflict with the at least one respective application associated with other groups of the plurality of groups.

In accordance with a fourth broad aspect, the invention provides a memory for storing data for access by an application program being executed on a data processing system under a secure environment in which a plurality of applications require shared resources. The memory has a data structure stored in the memory. The data structure includes information resident in a database used by the application program and includes, for each group of a plurality of groups, a plurality of first data objects containing information associating at least one respective application of the plurality of applications with the group; and a plurality of second data objects associating a respective resource allocation with the group for the at least one respective application associated with the group to allow the at least one respective application associated with the group to be executed on the computer system according to the respective resource allocation without conflict with the at least one respective application associated with other groups of the plurality of groups.

In accordance with a fifth broad aspect, the invention provides a method for a program to interact with a user to establish a secure environment for executing, on a computer system, a plurality of applications that require shared resources. The method involves displaying the plurality of applications and, for each group of a plurality of groups, receiving a selection associating at least one respective application of the plurality of applications with the group to associate a respective resource allocation with the group for

the at least one respective application associated with the group and allow the at least one respective application associated with the group to be executed on the computer system according the respective resource allocation without conflict with the at least one respective application associated with other groups of the plurality of groups.

In accordance with a sixth broad aspect, the invention provides a system adapted to perform any of the above method steps.

In another aspect, the invention relates to an apparatus to automate the process of extracting an application set installed on a computer system in such a way as to create an entity suitable for use as a secure software application container. The process involves parsing the files installed on an existing system, extracting the application and system specific information that is associated with an application set and creating a set of files that can be used in a secure application set container.

Brief Description of the Drawings

Preferred embodiments of the invention will now be described with reference to the attached drawings in which:

Figure 1 is a schematic of computing platforms operable to interact with containers which are created by combining applications and system files, according to an embodiment of the invention;

Figure 2 is a schematic of the computing platform of Figure 1 showing application sets being a collection of one or more software applications encompassed in a secure application container; and

Figure 3 is a flow diagram showing how application sets are created, according to another embodiment of the invention.

Detailed Description of the Preferred Embodiments

The following definitions are used herein:

- Computing platform: A computer system with a single instance of a fully functional operating system installed is referred to as a computing platform.

- Container: An aggregate of all files required to successfully execute a set of software applications on a computing platform is referred to as a container.

- Secure application container: An environment where each application set appears to have individual control of some critical system resources and/or where data within each application set is insulated from effects of other application sets is referred to as a secure application container.

- Consolidation. The ability to support multiple, possibly conflicting, sets of software applications on a single computing platform is referred to as consolidation.

Containers

Containers are created through the aggregation of application specific files.

A container contains application and data files for applications that provide a specific service. Examples of specific services include but are not limited to CRM (Customer Relation Management) tools, Accounting, and Inventory.

Referring to Figure 1, shown is a schematic of computing platforms operable to interact with containers that are created by combining applications and system files, according to an embodiment of the invention. A container in Figure 1 combines un-installed application files on a storage medium or network, application files installed on a computer, and system files. The container is an aggregate of all files required to successfully execute a set of software applications on a computing platform.

In embodiments of the invention, containers are created by combining application files with none or more system files. The containers are output to a file storage medium and installed on the computing platforms from the file storage medium. Each container contains application and data files for applications that together provide a specific service. The containers are created using, for example, Linux, Windows and Unix systems and preferably programmed in C and C++; however, the invention is not limited to these operating systems and programming languages and other operating systems and programming language may be used.

An application set is a set of one or more software applications that support a specific service.

As shown in Figure 1, application files are combined with system files to create a composite (or container) of the files required to support a fully functional software application.

Application specific files come from, for example, one of two sources:

- A distribution supplied by the creator of an application. These application specific files are often provided on a removable storage medium or downloaded in a packaged form.

- A computing platform in which the application specific files have been installed to support a fully functional software application and from which the application specific files are extracted.

The above two sources are used to categorize the applications as applications which have not yet been installed and applications which have already been installed and containers are created for both types of applications.

In some embodiments of the invention, system files are combined with application files as part of the aggregation process. These may include but are not limited to certain libraries, configuration files, and data sources.

Once a container is created it is placed in a file storage medium.

Secure Application Containers

In some cases, Applications from a container which contains application and data files for a specific service may need to share resources with Applications from a container which contains application and data files for another specific service. Some control over the shared resources may be required, for example, 1) for security purposes; 2) due to possible conflict arising from applications from different containers requiring a shared system resource, for example, the same port number; or 3) possible lack of resources on a system to accommodate all of the applications from different containers.

An environment where each application set appears to have individual control of critical system resources and/or where data from each application set is protected from effects of other application sets is referred to as a secure application container.

A secure application container is created on computing platforms for software applications that have been containerized. A secure application container creates an environment where application sets have unique access to critical system resources. Aspects of applications that may cause contention over shared resources are isolated in the secure application container.

In some embodiments of the invention, the secure application container is exported on a single instance of an operating system. This is different than prior art systems in which multiple, potentially conflicting application sets, are hosted on a single computing platform but with several operating systems. Current solutions (commonly called Virtual Machine (VM) technology) enable each application set to run within its own copy of an operating system. The net effect being, there are multiple instances of an operating system each running above a VM all on a single computing platform. For example, if eight application sets were hosted on a single computing platform using VM technology there would be eight operating systems sitting above eight VMs on the computing platform. The VM approach does not allow an operating system to utilize H/W resources directly, thus changing in significant ways the behavior of a system. In embodiments of the invention, the use of containerization allows an operating system to execute in it's native form on a computing platform.

For security purposes, data files within a secure application container are made inaccessible to applications from other containers. In some embodiments of the invention, this is achieved by providing the applications within secure application container with respective root file systems wherein the applications within a secure application container have a different root file system than those of applications within other secure application containers. Furthermore, data files and application files are made accessible only to users which are given permission to access applications within the secure application container. There are other security features such as: 1) providing each container with its own unique IP address that is independent of an IP address assigned to an underlying operating system on which the container is installed; 2) providing each container with a unique copy of a TCP/IP protocol stack for use by the container. This is unique copy of the protocol stack is used by the underlying operating system and application sets are not required to share resources associated the unique copy of the protocol stack with other application sets in other containers; and 3) provisioning each container with limits on consumption of hardware resources. In other words, each container is limited to an amount of CPU time, memory consumption, network bandwidth, disk storage and disk bandwidth. These limitations prevent an application in a container from doing harm to any other container or to the computing platform itself.

As discussed above, in prior art systems there may be conflicts between different applications in that the applications require the use of shared system resources. There are a number of potential resource conflicts within a computing platform. These could include, but are not limited to, conflicting uses of a common configuration file, conflicts over

firewall settings, any type of hardware device manipulation. By way of example, resource conflicts related to the use of network port numbers are described in some detail below as this is thought to be an easy to understand instance of a conflict situation over shared system resources.

An example of resource contention involves two applications on the same computing platform attempting to use the same port number. In such a case a conflict will result and one application will fail. Prior art systems allow for more than one IP (Internet Protocol) address to be used, through what is called aliasing, in conjunction with a single computing platform. This allows a computing platform to effectively provide more than one port for each service. For example, there can be two ports numbered 80 for web services on a single computing platform as each port 80 could be bound to a unique IP address. However, such services do not guarantee that, for example, two applications running on a computing platform will always be assigned different IP addresses. As such, although there may be more than one port available to provide a service, the two applications may still access the same IP address and a collision may still occur. To avoid collisions between applications from different containers, applications in different containers are given different IP (Internet Protocol) addresses. By associating respective IP addresses with the secure application containers, applications within a container are always assigned the same IP address. In this way, for a particular port number, an application is always assigned the same port.

Computing platforms have limited resources to accommodate different applications. For example, computing platforms have limits on their memory, CPU bandwidth, Disk size and bandwidth and Network bandwidth. In some embodiment of the

invention, provide is a capability to ensure that each container is able to use the resources that have been provisioned for it. In this manner each container is guaranteed to have sufficient resources to do work, but is not allowed to consume more than its allocation at the expense of resource guarantees for the other containers.

In some embodiments of the invention, workload management capabilities are provided that enable management of software applications by removing resource and file contention normally associated with combining multiple application sets on a single computing platform. Each secure application container contains information on resource requirements for applications within the container. Before a secure application container for a specific service is exported to a computing platform, the resources available on the platform are verified to determine whether the computing platform can accommodate applications within the secure application container. If one or more secure application containers are already installed on the computing platform, a check is performed to determine whether the computing platform has enough resources available to accommodate the secure application containers already installed and the secure application containers being exported. If there are enough resources available to accommodate the secure application container to be exported, it is exported and the resources are distributed between the secure application containers. When an application is executing, its resource consumption is then monitored and controlled to assure limits on consumption are not exceeded.

As discussed in United States Provisional Application entitled "USER MODE CRITICAL SYSTEM ELEMENTS AS SHARED LIBRARIES", which is incorporated herein by reference, the applications in the secure application containers execute in

user mode and a kernel portion of an operating system of a computing platform upon which the applications are installed executes in kernel mode. In conventional systems, there is a physical separation enforced by hardware between user mode and kernel mode and applications cannot run in kernel mode. System calls are used by the applications to make a transition from user mode to kernel mode. In some embodiments of the invention, resource usage of the applications within the secure application containers is monitored and system calls made by the applications are intercepted. When a system call made by an application executing in user mode is intercepted, the monitored resource usage for the application is compared to resource limits allocated to the application. If, for example, the system call requires the kernel to execute and cause the application to exceed the allocated resource limits the system call is prevented from being forwarded to the kernel, causing the application to pend, until the kernel can execute in kernel mode without the resource limits allocated to the application being exceeded. In this way control over shared resources is performed not only at an operating system level but also at a software level allowing further control over the shared resources.

The secure application containers assure separation between containers by providing each container with exclusive access to previously shared resources, such as, for example, a TCP/IP protocol stack, IP addresses and a root file system.

In some embodiments of the invention, multiple secure application containers exist above a single instance of an operating system and application sets are able to coexist inside secure application containers on the same computing platform.

With the use of the secure application containers potentially conflicting sets of applications are safely hosted on a single computing platform with a single instance of an operating system.

Container Extraction Tool

An extraction tool is used to discover the application specific files associated with given applications that are installed and working on a computing platform. The application specific files are extracted and used to create application sets as files on a file storage medium ready to install in secure application containers on computing platforms.

Software applications installed on computing platforms are extracted and converted into an intermediate format that allows the software applications to be installed in secure application containers on a remote compute platform. The process of extracting a software application installed on a fully functional computing platform is automated using a tool that executes on client systems which include, but are not limited to, Linux, Windows and Unix.

A database is used to track application specific files.

All application programs, application data and configuration information on a compute platform that are associated with specific installed software applications are identified.

The programs, data and configuration information are copied from the computing platform from which they originate and converted into an intermediate file format and stored on a

file storage medium. In one embodiment, a network storage device is used as the file storage medium. In some embodiments of the invention, the process of copying the programs, data and configuration information is achieved with the use of a user interface in which the application programs are displayed and selected for copying to a file storage medium using, for example, a drag and drop operation with a mouse, as described in United States Provisional Application entitled "Drag & Drop Application Management" which is incorporated herein by reference.

The programs, data and configuration information are then made available for installation into secure application containers on a remote computing platform.

In one embodiment of the invention, the container extraction tool is programmed in C++; however, the invention is not limited to the C++ programming language and in other embodiments of the invention other programming languages are used.

In some embodiments of the invention, containers are created from applications, at least one of which is an un-installed application. The un-installed applications is installed in a root file system.

Numerous modifications and variations of the present invention are possible in light of the above teachings. For example, embodiments of the invention have been described for implementation on computing platforms; however, the invention is not limited to implementations of computing platforms and in other embodiments of the invention there are implementations on computer systems. It is therefore to be understood that within the scope of the appended claims, the invention may be practiced otherwise than as specifically described herein.

50357-5

WE CLAIM:

1. A method of establishing a secure environment for executing, on a computer system, a plurality of applications that require shared resources, the method comprising:

for each group of a plurality of groups:

associating at least one respective application of the plurality of applications with the group; and

associating a respective resource allocation with the group for the at least one respective application associated with the group to allow the at least one respective application associated with the group to be executed on the computer system according to the respective resource allocation without conflict with the at least one respective application associated with other groups of the plurality of groups.

2. A method of executing the applications of claim 1 comprising, for each group of the plurality of groups executing on the computer system the at least one respective application associated with the group according to the respective resource allocation associated with the group.

3. A method according to claim 1 or 2 comprising, for each group of the plurality of groups, containerizing the at least one respective application associated with the group into a respective secure application container for the group, the respective secure application container being storable in a storage medium.

4. A method according to claim 3 wherein, for each group of the plurality of groups, the containerizing the at least one respective application associated with the group into a respective secure application container for the group comprises

containerizing application files for the at least one respective application associated with the group into the respective secure application container for the group.

5. A method according to claim 3 or 4 wherein, for each group of the plurality of groups, the containerizing the at least one respective application associated with the group into a respective secure application container for the group comprises containerizing application system calls for the at least one respective application associated with the group into the respective container for the group.

6. A method according to anyone of claims 1 to 5 wherein, for each group of the plurality of groups, the associating at least one respective application of the plurality of applications with the group comprises associating the at least one respective application of the plurality of applications with the group according to classes of service.

7. A method according to claim 6 wherein, for each group of the plurality of groups, the associating at least one respective application of the plurality of applications with the group comprises associating the at least one respective application of the plurality of applications with the group according to classes of service comprising specific application services.

8. A method according to claim 3 comprising exporting one or more of the respective secure application containers to a remote computer system.

9. A method according to claim 4 wherein each respective secure application container comprises data files for the at least one application within the respective secure application container, the method comprising making the data files within

one of the respective secure application containers inaccessible to the at least one respective application within another one of the respective secure application containers.

10. A method according to claim 9 wherein the making the data files within one of the respective secure application containers inaccessible to the at least one respective application within another one of the respective secure application containers comprises, for each respective secure application container providing the at least one respective application within the respective secure application container with a respective root file system.

11. A method according to claim 1 or 2 wherein, for each group of the plurality of groups, the associating a respective resource allocation with the group comprises associating a respective IP address for the group.

12. A method according to claim 1 or 2 wherein, for each group of the plurality of groups, the associating a respective resource allocation with the group comprises associating resource limits for the at least one respective application associated with the group.

13. A method according to claim 1 or 2 wherein, for each group of the plurality of groups, the associating a respective resource allocation with the group comprises associating resource limits comprising any one or more of limits on memory, CPU bandwidth, Disk size and bandwidth and Network bandwidth for the at least one respective application associated with the group.

14. A method according to claim 1 or 2 wherein each group of the plurality of groups comprises a secure application container comprising files related to the at least one

respective application associated with the group and, for each secure application container, the method further comprising:

determining whether resources available on the computer system can accommodate the respective resource allocation associated with the group comprising the secure application container; and

if the computer system can accommodate the respective resource allocation associated with the group comprising the secure application container, exporting the secure application container to the computer system.

15. A method according to claim 14 further comprising:

if one or more secure application containers of the secure application containers are already installed on the computer system, determining whether the computer system has enough resources available to accommodate the one or more secure application containers which are already installed and the secure application container being exported; and

if there are enough resources available, distributing the resources between the one or more secure application containers and the secure application container being exported to provide resource control.

16. A method according to claim 15 wherein the determining whether resources available on the computer system can accommodate the respective resource allocation comprises verifying whether the computer system supports any specific hardware required by the secure application container to be exported.

17. A method according to claim 2 wherein, for each group of the plurality of groups, the associating a respective

resource allocation with the group comprises associating resource limits for the at least one respective application associated with the group, the method further comprising during execution on the computer system:

monitoring resource usage of the at least one respective application associated with the group;

intercepting system calls, made by the at least one respective application associated with the group, from user mode to kernel mode;

comparing the monitored resource usage of the at least one respective application associated with the group with the resource limits; and

forwarding the system calls to a kernel on the basis of the comparison between the monitored resource usage and the resource limits.

18. A method according to claim 1 further comprising searching for application specific files associated with applications that are installed and working on computer systems.

19. A method according to claim 18 further comprising extracting the application specific files from the computer systems to create application sets ready to install in secure application containers on the computer system.

20. A method according to claim 19 further comprising converting the application specific files into an intermediate format that allows the applications that are installed and working on computer systems to be installed in secure application containers on the computer system.

21. A method according to claim 20 further comprising copying the application specific files, and related data and configuration information to a file storage medium.

22. A method according to claim 21 wherein the copying is achieved with the use of a user interface in which application programs are displayed and selected for copying to a storage medium.

23. A method according to claim 21 or 22 wherein the application specific files, and related data and configuration information are made available for installation into secure application containers on a remote computer system.

24. A method according to claim 1 further comprising installing at least one of the plurality of applications in a root file system.

25. A method of establishing a secure environment for executing, on a computer system, a plurality of applications that require shared resources, the method comprising:

for each secure application container of a plurality of secure application containers:

containerizing at least one respective application of the plurality of applications into the secure application container; and

associating a respective resource allocation with the secure application container for the at least one respective application containerized within the secure application container to allow the at least one respective application containerized within the secure application container to be executed on the computer system according to the respective resource allocation without conflict with the at least one

respective application containerized within other secure application containers of the plurality of secure application containers.

26. An article of manufacture comprising:

a computer usable medium having computer readable program code means embodied therein for establishing a secure environment for executing, on a computer system, a plurality of applications that require shared resources, the computer readable code means in said article of manufacture comprising:

for each group of a plurality of groups:

computer readable code means for associating at least one respective application of the plurality of applications with the group; and

computer readable code means for associating a respective resource allocation with the group for the at least one respective application associated with the group to allow the at least one respective application associated with the group to be executed on the computer system according to the respective resource allocation without conflict with the at least one respective application associated with other groups of the plurality of groups.

27. A memory for storing data for access by an application program being executed on a data processing system under a secure environment in which a plurality of applications require shared resources, the memory comprising:

a data structure stored in said memory, the data structure including information resident in a database used by said application program and including:

for each group of a plurality of groups:

a plurality of first data objects containing information associating at least one respective application of the plurality of applications with the group; and

a plurality of second data objects associating a respective resource allocation with the group for the at least one respective application associated with the group to allow the at least one respective application associated with the group to be executed on the computer system according to the respective resource allocation without conflict with the at least one respective application associated with other groups of the plurality of groups.

28. A method for a program to interact with a user to establish a secure environment for executing, on a computer system, a plurality of applications that require shared resources, the method comprising:

displaying the plurality of applications;

for each group of a plurality of groups;

receiving a selection associating at least one respective application of the plurality of applications with the group to associate a respective resource allocation with the group for the at least one respective application associated with the group and allow the at least one respective application associated with the group to be executed on the computer system according the respective resource allocation without conflict with the at least one respective application associated with other groups of the plurality of groups.

29. A system adapted to perform the method steps of any one of claims 1 to 25 and 28.

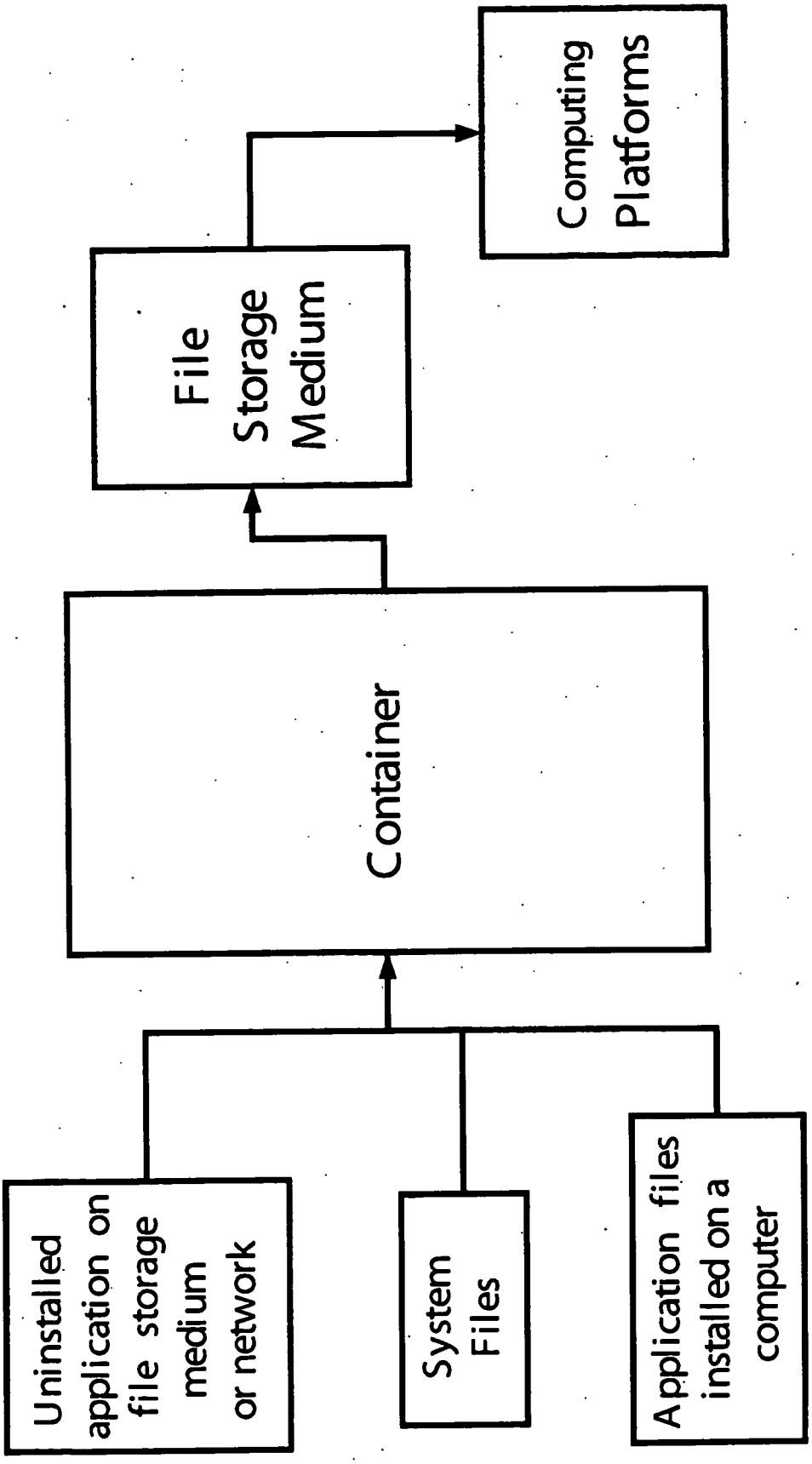


Figure 1.

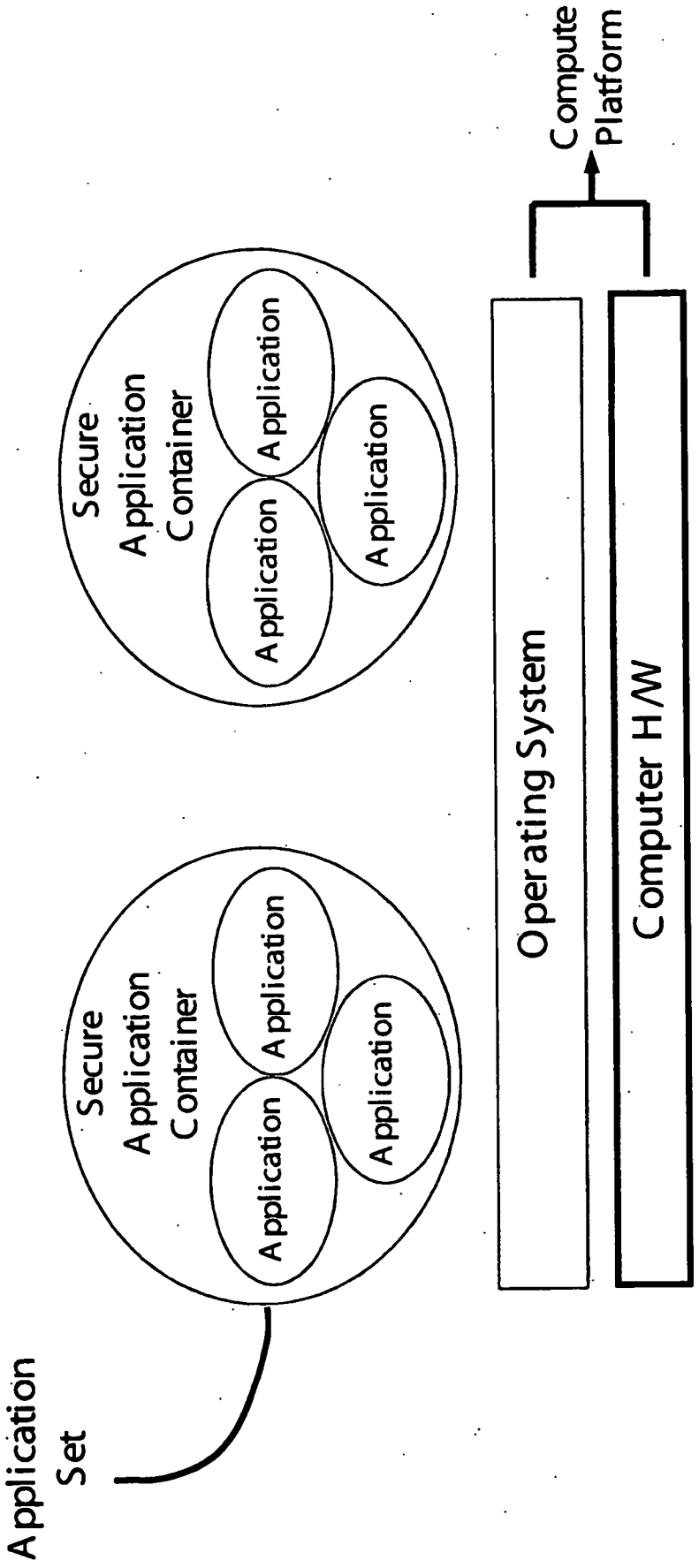


Figure 2.

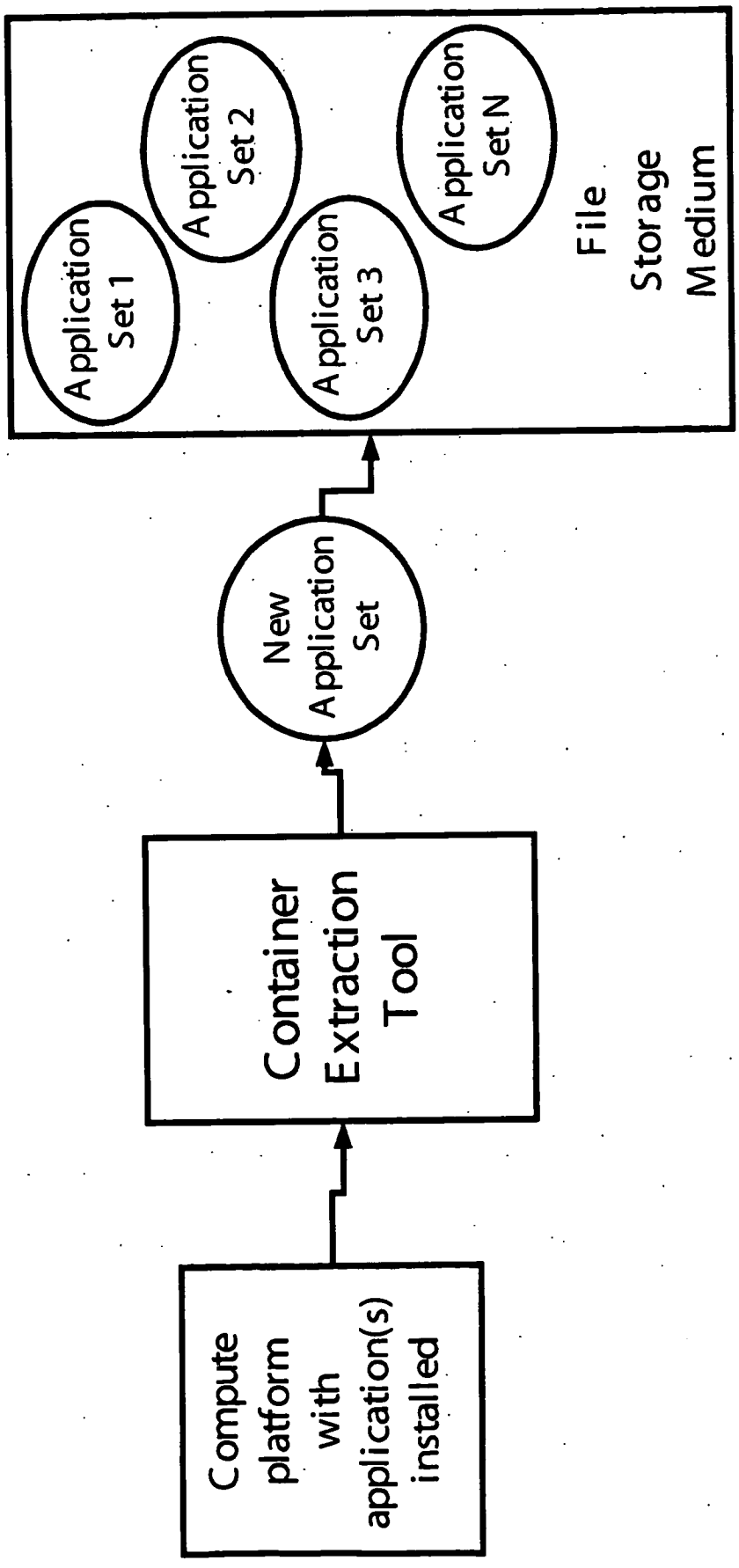


Figure 3.


Trilgence AE Console
File Configuration
System
Data Center
OFFLINE
ClearCase_User
ClearCase_Regi
Samba2_clone
working desktop
dell5p
dell8xxx
Bea0_cbt
Bea1_cbt
dell9p
samba1_cbt_wo
pts2
pts6
Bea2_dmp
Samba1_dmp
Details
Containers
Resources
CPU Utilization
100
50
0
9:19:43 AM
9:11:23 AM
Memory Utilization
3.4G
1.7G
0
9:19:43 AM
9:11:23 AM
Disk Throughput
22M
11M
0
9:19:43 AM
9:11:23 AM
Network Throughput
79
39
0
9:19:43 AM
9:11:23 AM
Legend
Bea0_cbt

Figure 14

Kraus & Weisert
Patent- und Rechtsanwälte
European Patent and Trademark Attorneys

Kraus & Weisert · Thomas-Wimmer-Ring 15 · 80539 München

European Patent Office
Erhardtstraße 27
80331 Munich

Dr. Walter Kraus (– 2002)
Dr.-Ing. Annekäte Weisert (– 2002)
Dr. Thomas Albrecht
Dipl.-Ing. Hans-Jörg Banzer
Dr. Holger Adam
Dr. Inge Hiebl
Dr. Claus Beckmann
Dr. Ferdinand Nielsen
Rechtsanwalt

Unser Zeichen / Our Ref.: 14738EP /nh

Bitte in der Antwort angeben
Please refer to in your reply

November 9, 2004

Re: European Patent Application 04 021 916.4
Trigence Corp.

Responsive to the Invitation to Remedy Deficiencies Pursuant to Rule 41(1) EPC:

Herewith a new copy of Figure 14 is filed. It is respectfully submitted that a better quality of the copy of Figure14 can not be provided, since said Figure shows a printout of a computer screen.

Should the deficiency not be corrected by the enclosed copy of Figure 14, a corresponding notification (possibly by phone) is politely requested. In this case, a formal drawing of Figure 14 will have to be prepared. However, said Figure will then be out of character, since it shows a printout of a computer screen.

Dipl.-Ing. Hans-Jörg Banzer
European Patent Attorney

Encl.:
New copy of Fig. 14

Thomas-Wimmer-Ring 15
80539 München

Tel. +49 (0)89 / 290 60-0
Fax +49 (0)89 / 290 60-111
office@kraus-weisert.de
www.kraus-weisert.de

HypoVereinsbank München
Kto.-Nr.: 667 340 080
BLZ: 700 202 70
IBAN: DE96 7002 0270 0667 3400 80
SWIFT: HYVEDEMM

Postbank München
Kto.-Nr.: 851 02-809
BLZ: 700 100 80
IBAN: DE67 7001 0080 0085 1028 09
SWIFT: PBNKDEFF

EPA/EPO/OEB
D-80298 München
☎ +49 89 2399-0
TX 523 656 epmu d
FAX +49 89 2399-4465

Europäisches Patentamt
Eingangsstelle

European Patent Office
Receiving Section

Office européen des brevets
Section de Dépôt

Banzer, Hans-Jörg, Dipl.-Ing.
Kraus & Weisert,
Thomas-Wimmer-Ring 15
80539 München
ALLEMAGNE

Datum/Date
04-11-2004

Zeichen/Ref./Réf.	Anmeldung Nr./Application No./Demande n°./Patent Nr ./Patent No./Brevet n°.
14738EP /nh	04021916.4-****

Anmelder/Applicant/Demandeur/Patentinhaber/Proprietor/Titulaire
Trigence Corp.

INVITATION TO REMEDY DEFICIENCIES PURSUANT TO RULE 41(1) EPC

The examination by the Receiving Section prescribed in Article 91(1)(b) to (d) EPC has disclosed that the above-mentioned European patent application contains the following deficiency (deficiencies):

(V) The APPLICATION DOCUMENTS do not correspond to the Implementing Regulations to the EPC (Rule 32(1)(2), Rule 35(2) to (10) and (14), Rule 36(1)(2) EPC).

(description: DE, claims: CL, drawings: DR)	DE	CL	DR
a) The text/printed matter on some*)/the sheets is not executed in dark, indelible colour and uniformity.	( )	( )	(V)
b) The document does not admit of direct reproduction.	( )	( )	(V)
c) Some*)/The sheets contain alterations/overwritings/ interlineations or too many erasures.	( )	( )	( )
d) The margins on some*)/the sheets are not as prescribed.	( )	( )	( )
e) Some*)/The sheets are not A4 size.	( )	( )	( )
f) The paper of some*)/the sheets is not white.	( )	( )	( )

*) sheet(s) see attached fig.14 .....

REGISTERED LETTER

Europäisches Patentamt
European Patent Office
Office européen des brevets

( ) The ABSTRACT has not been filed (Art. 91(1)(c) EPC).

( ) The REQUEST FOR GRANT has not been filed on the form as prescribed by the European Patent Office (Rule 26(1) EPC). A copy of this form may be found as an enclosure.

( ) The REQUEST FOR GRANT has not been signed by the representative/ (all) the applicant(s)(Rule 26(2)(i) EPC).

You are requested to remedy the deficiency (deficiencies) within TWO MONTHS after notification of this invitation.

If the specified deficiency (deficiencies) is (are) not remedied in due time, the European patent application will be refused (Art. 91(3) EPC).

The description, claims and drawings may be amended only to an extent sufficient to remedy the deficieny (deficiencies) specified above.

RECEIVING SECTION

Assimidou, Anna

Anmeldung Nr./Application No./Demande n°./Patent Nr./Patent No./Brevet n°.	Blatt/Page/Feuille
04021916.4	2

P.B.5818 - Patentlaan 2
2280 HV Rijswijk (ZH)
☎ +31 70 340 2040
TX 31651 epo nl
FAX +31 70 340 3016

Europäisches Patentamt | **European Patent Office** | **Office européen des brevets**

Zweigstelle in Den Haag Eingangsstelle | Branch at The Hague Receiving Section | Département à La Haye Section de Dépôt

Banzer, Hans-Jörg, Dipl.-Ing.
Kraus & Weisert,
Thomas-Wimmer-Ring 15
80539 München
ALLEMAGNE

Datum/Date
03-11-2004

Zeichen/Ref./Réf.	Anmeldung Nr./Application No./Demande n°./Patent Nr ./Patent No./Brevet n°.
14738EP /nh	04021916.4

Anmelder/Applicant/Demandeur/Patentinhaber/Proprietor/Titulaire
Trigence Corp.

NOTIFICATION PURSUANT TO RULE 42 (ARTICLE 91(2) EPC)
DEFICIENCIES CONCERNING DESIGNATION OF INVENTOR

Your attention is drawn to the following deficiency/deficiencies in relation to the designation of the inventor in the above-mentioned European patent application.

( ) Where the applicant is the sole inventor: The designation is missing in the request for grant (Rule 26(2)(k) EPC).

(V) Where the applicant is not the inventor or the sole inventor:
(V) A separate document containing the designation drawn up according to Rule 17(1) and Article 81 EPC has not been filed.
( ) The document containing the designation does not comply with Article 81 and Rule 17(1) EPC since
( ) the document is not signed.
( ) the document is not duly signed. Reasons: .....

.....

You are invited to remedy the deficiency/deficiencies:
(V) within 16 MONTHS after the date of filing or, if priority is claimed, after the earliest date of priority.
( ) within TWO MONTHS after notification of this invitation.
These time limits are non-extendable.

If you fail to do so, the application will be deemed to be withdrawn (Art. 91(5) EPC).

RECEIVING SECTION

Assimidou, Anna

Europäisches Patentamt • European Patent Office • Office européen des brevets

REGISTERED LETTER

EPO Form 1045 (03.00) 7002007 18/09/04

04021916.4 (FORI) 000

Abstract of the Disclosure

A system is disclosed having servers with operating systems that may differ, operating in disparate computing environments, wherein each server includes a processor and an operating system including a kernel a set of associated local system files compatible with the processor. This invention discloses a method of providing at least some of the servers in the system with secure, executable, applications related to a service, wherein the applications may be executed in a secure environment, wherein the applications each include an object executable by at least some of the different operating systems for performing a task related to the service. The method of this invention requires storing in memory accessible to at least some of the servers a plurality of secure containers of application software. Each container includes one or more of the executable applications and a set of associated system files required to execute the one or more applications, for use with a local kernel residing permanently on one of the servers. The set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems. The containers of application software exclude a kernel; and some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files resident on the server.

(Fig. 1)

What is claimed is:

1. In a system having a plurality of servers with operating systems that differ, operating in disparate computing environments, wherein each server includes a processor and an operating system including a kernel a set of associated local system files compatible with the processor, a method of providing at least some of the servers in the system with secure, executable, applications related to a service, wherein the applications may be executed in a secure environment, wherein the applications each include an object executable by at least some of the different operating systems for performing a task related to the service, the method comprising the steps of:

storing in memory accessible to at least some of the servers a plurality of secure containers of application software, each container comprising one or more of the executable applications and a set of associated system files required to execute the one or more applications, for use with a local kernel residing permanently on one of the servers; wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems, the containers of application software excluding a kernel, and wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files resident on the server prior to said storing step.

2. A method as defined in claim 1, wherein each container has an execution file associated therewith for starting the one or more applications, and wherein the execution file includes instructions related to an order in which executable applications within will be executed, and wherein in operation when applications are executed, applications within a container have no access to system files or applications in other containers or system files within the operating system during execution thereof if those applications or system files are read/write files.

3. A method as defined in claim 1 or claim 2 further comprising the step of pre-identifying applications and system files required for association with the one or more containers prior to

said storing step, and modifying at least some of the system files to provide an association with a container specific identity assigned to the container.

4. A method as defined in any one of claims 1, 2, or 3 comprising the step of assigning a unique associated identity to each of a plurality of the containers, wherein the identity includes at least one of IP address, host name, and MAC address.

5. A method as defined in any one of claims 1 through 4, wherein the one or more applications and associated system files are retrieved from a computer system having a plurality of secure containers.

6. A method as defined in any one of claims 2 through 5 wherein server information related to hardware resource usage including at least one of CPU memory, network bandwidth and disk allocation is associated with at least some of the containers prior to the applications within the containers being executed.

7. A method as defined in any of claims 2 through 6, wherein containers include files stored in network file storage, and parameters forming descriptors of containers stored in a separate location.

8. A method as defined in any of claims 1 through 7 comprising the step of creating containers prior to said step of storing containers in memory, wherein the step of creating containers comprises the steps of:

a) running an instance of a service on a server,

determining which files are being used, and,

copying applications and associated system files to memory;

or,

(b) using a skeleton set of system files as a container starting point and installing applications into that set of files.

9. A method as defined in any of claims 1 to 8 further comprising the step of installing a service on a target server selected from one of the plurality of servers, wherein the step of installing the service includes the steps of:
using a graphical user interface, associating a unique icon representing a service with an unique icon representing a server for hosting applications related to the service and for executing the service, so as to cause the applications to be distributed to, and installed on the target server.

10. A method as defined in claim 9 wherein the target server and the graphical user interface are at remote locations, or wherein the graphical user interface is installed on a computing platform, and wherein the computing platform is a different computing platform than the target server.

11. A method as defined in any of claims 9 or 10, wherein the step of associating includes the step of relatively moving the unique icon representing the service to the unique icon representing a server.

12. A method as defined in any of claims 9 through 11 further comprising the step of: de-installing a service from a server, comprising the steps of:
displaying the unique icon representing the service;
displaying the unique server icon representing the server on which the service is installed;
and utilizing the icon representing the service and the icon representing the server to initiating the de-installation of the selected software application from the selected server.

13. A method according to any of claims 9 through 12 further comprising the step of separating the icon representing the service from the icon representing the server.

14. A method according to claim 13 further comprising the step of copying data file changes specific to the service back to a storage medium from which the data file changes originated prior to installation.

15. A computing system for performing a plurality of tasks each comprising a plurality of processes comprising:

a plurality of secure stored containers of associated files accessible to, and for execution on, one or more servers, each container being mutually exclusive of the other, such that read/write files within a container cannot be shared with other containers, each container of files having its own unique identity associated therewith, said identity comprising at least one of an IP address, a host name, and a Mac_address; wherein, the plurality of files within each of the plurality of containers comprise one or more application programs including one or more processes, and associated system files for use in executing the one or more processes, each container having its own execution file associated therewith for starting one or more applications, in operation, each container utilizing a kernel resident on the server and wherein each container exclusively uses a kernel in an underlying operation system in which it is running and is absent its own kernel; and,

a run time module for monitoring system calls from applications associated with one or more containers and for providing control of the one or more applications.

16. A computing system as defined in claim 15, further comprising a scheduler comprising values related to an allotted time in which processes within a container may utilize predetermined resources and, wherein the run time module includes an intercepting module associated with the plurality of containers for intercepting system calls from any of the plurality of containers and for providing values alternate to values the kernel would have assigned in response to the system calls, so that the containers can run independently of one another without contention, in a secure manner, the values corresponding to at least one of the IP address, the host name and the Mac_Address.

17. A computing system as defined in claim 16, wherein the run time module performs:
monitoring resource usage of applications executing;
intercepting system calls to kernel mode, made by the at least one respective application within a container, from user mode to kernel mode;

comparing the monitored resource usage of the at least one respective application with the resource limits; and,

forwarding the system calls to a kernel on the basis of the comparison between the monitored resource usage and the resource limits.

5

SYSTEM FOR CONTAINERIZATION OF APPLICATION SETS

Field of the Invention

The present invention relates to management and deployment of server applications in computer systems.

Background of the Invention

Computer systems are designed in such a way that application programs share common resources. It is traditionally the task of an operating system to provide a mechanism to safely and effectively control access to shared resources required by application programs. This is the foundation of multi-tasking systems that allow multiple disparate applications to co-exist on a single computer system.

The current state of the art creates an environment where a collection of applications, each designed for a distinct function, must be separated with each application installed on an individual computer system.

In some instances this separation is necessitated by conflict over shared resources, such as network port numbers that would otherwise occur. In other situations the separation is necessitated by the requirement to securely separate data such as files contained on disk-based storage and/or applications between disparate users.

In yet other situations, the separation is driven by the reality that certain applications require a specific version of operating system facilities and as such will not co-exist with applications that require another version.

As computer system architecture is applied to support specific services, it inevitably requires that separate systems be deployed for different sets of applications required to perform and or support specific services. This fact, coupled with increased demand for support of additional application sets, results in a significant increase in the number of computer systems being

deployed. Such deployment makes it quite costly to manage the number of systems required to support several applications.

There are existing solutions that address the single use nature of computer systems. These solutions each have limitations, some of which this invention will address. Virtual Machine technology, pioneered by VmWare, offers the ability for multiple application/operating system images to effectively co-exist on a single compute platform. The key difference between the Virtual Machine approach and the approach described herein is that in the former an operating system, including files and a kernel, must be deployed for each application while the latter only requires one operating system regardless of the number of application containers deployed. The Virtual Machine approach imposes significant performance overhead. Moreover, it does nothing to alleviate the requirement that an operating system must be licensed, managed and maintained for each application. The invention described herein offers the ability for applications to more effectively share a common compute platform, and also allow applications to be easily moved between platforms, without the requirement for a separate and distinct operating system for each application.

A product offered by Softricity, called SoftGrid®, offers what is described as Application Virtualization. This product provides a degree of separation of an application from the underlying operating system. Unlike Virtual Machine technology a separate operating system is not required for each application. The SoftGrid® product does not isolate applications into distinct environments. Applications executing within a SoftGrid® environment don't possess a unique identity.

This invention provides a solution whereby a plurality of services can conveniently be installed on one or more servers in a cost effective and secure manner.

The following definitions are used herein:

•Disparate computing environments: Environments where computers are stand-alone or where there are plural computers and where they are unrelated.

•Computing platform: A computer system with a single instance of a fully functional operating system installed is referred to as a computing platform.

•Container: An aggregate of files required to successfully execute a set of software applications on a computing platform is referred to as a container. A container is not a physical container but a grouping of associated files, which may be stored in a plurality of different locations that is to be accessible to, and for execution on, one or more servers. Each container for use on a server is mutually exclusive of the other containers, such that read/write files within a container cannot be shared with other containers. The term "within a container", used within this specification, is to mean "associated with a container". A container comprises one or more application programs including one or more processes, and associated system files for use in executing the one or more processes; but containers do not comprise a kernel; each container has its own execution file associated therewith for starting one or more applications. In operation, each container utilizes a kernel resident on the server that is part of the operating system (OS) the container is running under to execute its applications.

•Secure application container: An environment where each application set appears to have individual control of some critical system resources and/or where data within each application set is insulated from effects of other application sets is referred to as a secure application container.

•Consolidation: The ability to support multiple, possibly conflicting, sets of software applications on a single computing platform is referred to as consolidation.

•System files: System files are files provided within an operating system and which are available to applications as shared libraries and configuration files.

By way of example, Linux Apache uses the following shared libraries, supplied by the OS distribution, which are "system" files.

/usr/lib/libz.so.1
/lib/libssl.so.2
/lib/libcrypto.so.2
/usr/lib/libaprutil.so.0
/usr/lib/libgdbm.so.2
/lib/libdb-4.0.so
/usr/lib/libexpat.so.0
/usr/lib/libapr.so.0
/lib/i686/libm.so.6
/lib/libcrypt.so.1
/lib/libnsl.so.1
/lib/libdl.so.2
/lib/i686/libpthread.so.0
/lib/i686/libc.so.6
/lib/ld-linux.so.2

Apache uses the following configuration files, also provided with the OS distribution:

/etc/hosts
/etc/httpd/conf
/etc/httpd/conf.d
/etc/httpd/logs
/etc/httpd/modules
/etc/httpd/run

By way of example, together these shared library files and configuration files form system files provided by the operating system. There may be any number of other files included as system files. Additional files might be included, for example, to support maintenance activities or to start other network services to be associated with a container.

Summary of the Invention

The present invention provides a method as defined in claim 1 and a computing system as defined in claim 15. The dependent claims define respective preferred or advantageous embodiments.

In a system having a plurality of servers with operating systems that differ, operating in disparate computing environments, wherein each server includes a processor and an operating system including a kernel a set of associated local system files compatible with the processor, a method is provided for providing at least some of the servers in the system with secure, executable, applications related to a service, wherein the applications may be executed in a secure environment, wherein the applications each include an object executable by at least some of the different operating systems for performing a task related to the service. The method comprises the steps of:

storing in memory accessible to at least some of the servers a plurality of secure containers of application software, each container comprising one or more of the executable applications and a set of associated system files required to execute the one or more applications, for use with a local kernel residing permanently on one of the servers; wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems, the containers of application software excluding a kernel, and wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files resident on the server prior to said storing step.

In accordance with another aspect of the invention a computer system is provided for performing a plurality of tasks each comprising a plurality of processes comprising:

a plurality of secure stored containers of associated files accessible to, and for execution on, one or more servers, each container being mutually exclusive of the other, such that read/write files within a container cannot be shared with other containers, each container of files having its own unique identity associated therewith, said identity comprising at least one of an IP address, a host name, and a MAC address; wherein, the plurality of files within each of the plurality of containers comprise one or more application programs including one or more processes, and associated system files for use in executing the one or more

processes, each container having its own execution file associated therewith for starting one or more applications, in operation, each container utilizing a kernel resident on the server and wherein each container exclusively uses a kernel in an underlying operation system in which it is running and is absent its own kernel; and, a run time module for monitoring system calls from applications associated with one or more containers and for providing control of the one or more applications.

In accordance with a broad aspect, the invention provides a method of establishing a secure environment for executing, on a computer system, a plurality of applications that require shared resources. The method involves, associating one or more respective software applications and required system files with a container. A plurality of containers have these containerized files within.

Containers residing on or associated with a respective server each have a resource allocation associated therewith to allow the at least one respective application or a plurality of applications residing in the container to be executed on the computer system according to the respective resource allocation without conflict with other applications in other containers which have their given set of resources and settings.

Containers, and therefore the applications within containers, are provided with a secure environment from which to execute. In some embodiments of the invention, each group of applications is provided with a secure storage medium.

In some embodiments of the invention the method involves containerizing the at least one respective application associated respective secure application container, the respective secure application container being storable in a storage medium.

In some embodiments of the invention, groups of applications are containerized into a single container for creating a single service. By way of example applications such as Apache, MySql and PHP may all be grouped in a single container for supporting a single service, such as a web based human resource or customer management type of service.

In some embodiments of the invention, the method involves exporting one or more of the respective secure application containers to a remote computer system.

In an embodiment of the invention, each respective secure application container has data files for the at least one application within the respective secure application container.

The method involves making the data files within one of the respective secure application containers inaccessible to any respective applications within another one of the secure application containers.

In embodiments of the invention, all applications within a given container have their own common root file system exclusive thereto and unique from other containers and from the operating system's root file system.

In embodiments of the invention, a group of applications within a single container share a same IP address, unique to that container.

Hence in some embodiments of the invention a method is provided for associating a respective IP address for a group of applications within a container.

In some embodiments of the invention, for each container of applications, the method involves associating resource limits for all applications within the container.

In some embodiments of the invention, for each group of the plurality of groups of applications, for example, for each container of applications and system files, the method involves associating resource limits comprising any one or more of limits on memory, CPU bandwidth, Disk size and bandwidth and Network bandwidth for the at least one respective application associated with the group.

For each secure application container, the method involves determining whether resources available on the computer system can accommodate the respective resource allocation associated with the group of applications comprising the secure application container.

By way of example, when a container is to be placed on a platform comprising a computer and an operating system (OS) a check is done to ensure that the computer can accommodate the resources required for the container. Care is exercised to ensure that the installation of a container will not overload the computer. For example, if the computer is using 80% of its CPU capacity and installing the container will increase its capacity past 90% the container is not installed.

If the computer system can accommodate the respective resource allocation associated with the group of applications forming the secure application container, the secure application container is exported to the computer system.

Furthermore, in another embodiment, if one or more secure application containers are already installed on the computer system, the method involves determining whether the computer system has enough resources available to accommodate the one or more secure application containers are already installed in addition to the secure application container being exported.

If there are enough resources available, the resources are distributed between the one or more secure application containers and the secure application container being exported to provide resource control.

In some embodiments of the invention, in determining whether resources available on the computer system to accommodate the respective resource allocation, the method involves verifying whether the computer system supports any specific hardware required by the secure application container to be exported. Therefore, a determination can be made as whether any specific hardware requirements of the applications within a container are supported by the server into which the container is being installed. This is somewhat similar to the

abovementioned step wherein a determination is made as to whether the server has sufficient resources to support a container to be installed.

In some embodiments of the invention, for each group of applications within a container, in associating a respective resource allocation with the group, the method involves associating resource limits for the at least one respective application associated with the group.

More particularly, the method also involves, during execution on the computer system:
monitoring resource usage of the at least one respective application associated with the group;
intercepting system calls to kernel mode, made by the at least one respective application associated with the group of applications from user mode to kernel mode;
comparing the monitored resource usage of the at least one respective application associated with the group with the resource limits;
and forwarding the system calls to a kernel on the basis of the comparison between the monitored resource usage and the resource limits.

The above steps define that the resource limit checks are done by means of a system call intercept, which is, by definition, a hardware mechanism where an application in user mode transitions to kernel code executing in a protected mode, i.e. kernel mode. Therefore, the invention provides a mechanism whereby certain, but not all system calls are intercepted; and wherein operations specific to the needs of a particular container are performed, for example, checks determines if limits may have been exceeded, and then allows the original system to proceed.

Furthermore, in some instances a modification may be made to what is returned to the application; for example, when a system call is made to retrieve the IP address or hostname. The system in accordance with an embodiment of this invention may choose to return a container specific value as opposed to the value that would have otherwise been normally returned by the kernel. This intervention is termed "spoofing" and will be explained in greater detail within the disclosure.

Brief Description of the Drawings

Exemplary embodiments may be envisaged without departing from the spirit and scope of the invention.

Figure 1 is a schematic diagram illustrating a containerized system in accordance with an embodiment of this invention.

Figure 2 is a schematic diagram illustrating a system wherein secure containers of applications and system files are installed on a server in accordance with an embodiment of the invention.

Figure 3 is a pictorial diagram illustrating the creation of a container.

Figure 4 is a diagram illustrating how a container is assigned a unique identity such as IP address, Hostname, and MAC address and introduces the "kernel module" which is used to handle system calls.

Figure 5 is a diagram similar to Figure 4, wherein additional configuration data is provided.

Figure 6 is a diagram illustrating a system wherein the containers are secure containers.

Figure 7 is a diagram introducing the sequencing or order in which applications within a container are executed.

Figure 8 is a diagram illustrating the relationship between the storage of container system files and container configuration data for a plurality of secure containers which may be run on a particular computer platform.

Figure 9 is a diagram that illustrates the installation of a container on a server.

Figure 10 is a diagram that illustrates the monitoring of a number of applications and state information.

Figure 11 is a diagram which shows that the container creation process includes the steps to install the container and validate that applications associated with the container are functioning properly.

Figure 12 is a schematic diagram showing the operation of the invention, according to an embodiment of the invention.

Figure 13 is a schematic diagram showing software application icons collected in a centralized file storage medium and a remote computing platform icon, according to another embodiment of the invention.

Figure 14 is a screen snapshot of an implementation according to the schematic of Figure 13.

Figure 15 is a flow chart of a method of initializing icons of Figure 14.

Figure 16 is a flow chart of a method of installing a software application on a computing platform in response to a drag and drop operation of Figure 14; and,

Figure 17 is a flow chart of a method of de-installing a software application from a computing platform in response to a drag and drop operation of Figure 13.

Detailed Description

Turning now to Figure 1, a system is shown having a plurality of servers 10a, 10b. Each server includes a processor and an independent operating system. Each operating system includes a kernel 12, hardware 13 in the form of a processor and a set of associated system files, not shown. Each server with an operating system installed is capable of executing a number of application programs. Unlike typical systems, containerized applications are shown on each server having application programs 11a, 11b and related system files 11c.

These system files are used in place of system files such as library and configuration files present typically used in conjunction with the execution of applications.

Figure 2 illustrates a system in accordance with the invention in which multiple applications 21a, 21b, 21c, 21d, 21e, and 21f can execute in a secure environment on a single server. This is made possible by associating applications with secure containers 20a, 20b and 20c. Applications are segregated and execute with their own copies of files and with a unique identity. In this manner multiple applications may contend for common resources and utilize different versions of system files. Moreover, applications executing within a secure container can co-exist with applications that are not associated with a container, but which are associated with the underlying operating system.

Containers are created through the aggregation of application specific files. A container contains application and data files for applications that provide a specific service. Examples of specific services include but are not limited to CRM (Customer Relation Management) tools, Accounting, and Inventory.

The Secure application containers 20a, 20b and 20c are shown in Figure 2 in which each combines un-installed application files on a storage medium or network, application files installed on a computer, and system files. The container is an aggregate of all files required to successfully execute a set of software applications on a computing platform. In embodiments of the invention, containers are created by combining application files with system files.

The containers are output to a file storage medium and installed on the computing platforms from the file storage medium. Each container contains application and data files for applications that together provide a specific service. The containers are created using, for example, Linux, Windows and Unix systems and preferably programmed in C and C++; however, the invention is not limited to these operating systems and programming languages and other operating systems and programming language may be used. An application set is a set of one or more software applications that support a specific service. As shown in the

figures, which follow, application files are combined with system files to create a composite or container of the files required to support a fully functional software application.

Referring now to Figure 3 the creation of a container is illustrated from the files used by a specific application and user defined configuration information. The required files can be captured from an existing computer platform 32 where an application of interest is currently installed or the files can be extracted from install media or package files 30. Two methods of container creation will be described hereafter.

Figure 4 illustrates a system having two secure containers 20a 20b, each provided with a unique identity. This allows, for example, other applications to locate a containerized service in a consistent manner independent of which computer platform the containerized service is located on. Container configuration data, collected from a user or user-defined program, is passed to services resident on a computer platform 40 hosting containers. One embodiment shows these services implemented in a kernel module 43. The configuration information is transferred one time when the container is installed on a platform. The type of information required in order to provide an application associated with a container a unique identity includes items such as an IP address, MAC address and hostname. As applications execute within a container 20a, 20b environment system calls are intercepted, by the kernel module 43 passing through services installed as part of this invention, before being passed on to the kernel. This system call intercept mechanism allows applications to be provided with values that are container specific rather than values that would otherwise be returned from the kernel by "spoofing" the system.

As introduced heretofore, "spoofing" is invoked when a system call is made. Instead of returning values ordinarily provided by the operating system's kernel a module in accordance with this invention intervenes and returns container specific values to the application making the system call. By way of example, when an application makes the 'uname' system call, the kernel returns values for hostname, OS version, date, time and processor type. The software module in accordance with this invention intercepts the system call and causes the application to see kernel defined values for date, time and processor type; however, the

hostname value is purposely changed to one that is container specific. Thus, the software has 'spoofed' the 'uname' system call to return a container specific hostname value rather than the value the kernel would ordinarily return. This same "spoofing" mechanism applies to system calls that return values such as MAC address, etc. A similar procedure exists for network related system calls, such as bind. For other system calls, such as fork and exit (create and delete a new process) the software does not alter what is returned to the application from the kernel; however, the system records that an operation has occurred, which results in the system call intercept of fork not performing any spoofing. The fork call is intercepted for purposes of tracking container-associated activity.

By way of example, if a process within a container creates a new process, by making a fork system call, the new process would be recorded as a part of the container that created it.

System calls are intercepted to perform the following services:
tracking applications, for example a tracking of which applications are in and out of a specific container;
spoofing particular values returned by the kernel;
applying resource checks and imposing resource limits;
monitoring whether an application is executing as expected, retrieving application parameters; and, which applications are being used, by who and for how long; and,
retrieving more complex application information including information related to which files have been used, which files have been written to, which sockets have been used and information related to socket usage, such as the amount of data transferred through a given socket connection.

Container configuration includes the definition of hardware resource usage, as depicted in Figure 5 including the amount of CPU, memory, network bandwidth and disk usage required to support the applications to be executed in association with the container 20a or 20b. These values can be used to determine resource requirements for a given compute platform 40. They can also be used to limit the amount of hardware resources allocated to applications associated with a container. Values for hardware resources can be defined by a user 51 when

a container is created. They can be modified after container creation. It is also possible for container runtime services to detect the amount of resources utilized by applications associated with a container. In the automatic detection method values for hardware resources are updated when the container is removed from a compute platform. The user-defined values can be combined with auto-detected values as needed. In this model a user defines values that are then modified by measured values and updated upon container removal.

It can be seen from Figure 6 that each application executing associated with a container 20a or 20b, or contained within a container 20a or 20b, is able to accesses files that are dedicated to the container. Applications with a container association, that is, applications contained within a container, are not able to access the files provided with the underlying operating system of the compute platform 40 upon which they execute. Moreover, applications with a container 20a association are not able to access the files contained in another container 20b.

When a container 20a or 20b is installed specific applications are started, as is shown in Figure 7 and container configuration information defines how applications 71, 72 are started. This includes the definition of a first program to start when a container is installed on a compute platform 40. The default condition, if not customized for a specific container, will start a script that in turn runs other scripts. A script associated with each application is provided in the container file system. The controller script 73, delineating the first application started in the container, runs each application specific script in turn. This allows for any number of applications to be started with a container association.

Special handling is required to start the first application such that it is associated with a container. Subsequent applications and processes created, as a result of the first application, will be automatically associated with the container. The automatic association is done through tracking mechanisms based on system call intercept. These are contained in a kernel module 43 in one embodiment of the invention. For example, fork, exit and wait system calls are intercepted such that container association can be applied to applications.

The first application started when a container is installed, is associated with the container by informing the system call intercept capabilities, embodied in the kernel module, shown in Figure 7, that the current process is part of the container. Then, the application 71 is started by executing the application as part of the current process. In this way, the first application is started in the context of a process that has been associated with the container.

A container has a physical presence when not installed on a compute platform 40. It is described as residing on a network accessible repository. As is shown in Figure 8, a container has a physical presence in two locations 82, 84; or stated differently, the files associated with a container have a physical presence in two locations 82, 84. The files used by applications associated with a container reside on a network file server 82. Container configuration is managed by a controller. The configuration state is organized in a database 84 used by the controller. A container then consists of the files used by applications associated with the container and configuration information. Configuration information includes those values which provide a container with a unique identity, for example a unique IP address, MAC address, name, etc., and which describe how a container is installed, starts a first application and shuts down applications when removed.

In a first embodiment all files are exclusive. That is, each container has a separate physical copy of the files required by applications associated with the container. In future embodiments any files that is read-only will be shared between containers.

When a container is installed on a compute platform the files used by applications associated with the container and container specific configuration information elements are combined. Figure 9 shows that the files 82a, 84a are either copied, physically placed on storage local to the compute platform 40, or they are mounted from a network file storage server. When container files are mounted no copy is employed.

Configuration information is used to direct how the container is installed. This includes operations such as describing the first application to be started, mount the container files, if needed, copy files, if needed and create an IP address alias.

Container information is also used to provide the container with a unique identity, such as IP address, MAC address and name. Identity information is passed to the system call intercept service, shown as part of a kernel module 43 in Figure 9.

As the software application associated with or contained in a container is executed it is monitored through the use of system call intercept. Figure 10 shows that a number of operations are monitored. State information relative to application behavior is stored in data structures managed by the system call intercept capabilities. The information gathered in this manner is used to track which processes are associated with a container, their aggregate hardware resource usage and to control specific values returned to the application. Hardware resource usage includes CPU time, memory, file activity and network bandwidth.

Figure 3 illustrates container creation. The result of container creation is the collection of files required by applications associated with the container and the assembly of container specific configuration information. Container files include both those files provided by an operating system distribution and those files provided by an application install media or package file.

These files can be obtained by using a compute platform upon which the application of interest has been installed. In this case the files are copied from the existing platform. Alternatively, a process where the files from the relative operating system distribution are used as the container files, the container is installed on a compute platform an then application specific files are applied from applicable install media or package file. The result in either case is the collection of all files needed by applications associated with the container onto a network accessible repository.

Container specific configuration information is assembled from user input. The input can be obtained from forms in a user interface, or programmatically through scripting mechanisms.

Two methods of container creation are provided and either of the two can be utilized, depending upon particular circumstances. In a first method of container creation a "skeleton" file system is used. This is a copy of the system files provided with a given operating system (OS) distribution. The skeleton file system is placed on network storage, accessible to other servers. The skeleton file system includes the OS provided system files before an application is installed. A container is created from this skeleton file system. Subsequently, a user logs into the container and installs applications as needed. Most installations use a service called ssh to login to the system which is used to log into a container.

Referring once again to Figure 3, applications may come from third party installation media on CDROM, package files 30, or as downloads from a web site, etc. Once installed in the container the applications become unique to the container in the manner described way that has been described heretofore.

The second method employed to create a container file system uses an existing server, for example a computer or server already installed in a data center. This is also shown in Figure 3. The applications of interest may have been installed on an existing server before the introduction of the software and data structures in accordance with this invention. Files are copied from the existing server 32, including system files and application specific files to network storage. Once the files are copied from the existing server a container is created from those files.

The description of the two methods above differs in that in one instance the container creation begins with the system files only. The container is created from the system files and then the applications and required files are added and later installed. In the second instance, which is simpler, both system and application files are retrieved, together, from an existing server, and then the container is created. In this manner, the container file system comprises the application and system files.

Once a set of files that are retrieved for creating a container, for example system files only or system and application files, a subset of the system files are modified. The changes made to this subset are quite diverse. Some examples of these changes are:
modifying the contents a file to add a container specific IP address;
modifying the contents of a directory to define start scripts that should be executed;
modifying the contents of a file to define which mount points will relate to a container;
removing certain hardware specific files that are not relevant in the container context, etc., dependent upon requirements.

In some embodiments of the invention, the method involves copying the application specific files, and related data and configuration information to a file storage medium.

This may be achieved with the use of a user interface in which application programs are displayed and selected for copying to a storage medium. In some embodiments of the invention, the application specific files, and related data and configuration information are made available for installation into secure application containers on a remote computer system.

In some embodiments of the invention, the method involves installing at least one of the pluralities of applications in a container's own root file system.

In summary, the invention involves combining and installing at least one application along with system files or a root file system to create a container file system. A container is then created from a container file system and container configuration parameters.

A resource allocation is associated with the secure application container for at least one respective application containerized within the secure application container to allow the at least one respective application containerized within the secure application container to be executed on the computer system without conflict with other applications containerized within other secure application containers.

Thus, a container has an allocation of resources associated with it to allow the application within the container to be executed without contention.

This allocation of resources is made during the container configuration as a step in creating the container. An active check for resource allocation against resources available is performed. Containerized applications may run together within a container; and, non-containerized applications may run outside of a container context on a same computer platform.

Drag and Drop Application Management

Another aspect of this relates to software that manages the operation of a data center.

There has been an unprecedented proliferation of computer systems in the business enterprise sector. This has been a response to an increase in the number and changing nature of software applications supporting key business processes. Management of computer systems required to support these applications has become an expensive proposition. This is partially due to the fact that each of the software applications are in most cases hosted on an independent computing platform or server. The result is an increase in the number of systems to manage.

An aspect of this invention provides a method of installing a software application on a computing platform. The terms computing platform, server and computer are used interchangeably throughout this specification. The method in accordance with this aspect of the invention includes displaying a software application icon representing the software application and a computing platform icon representing the computing platform.

In operation, a selection of the software application for installation is initiated using the software application icon; and a selection of the computing platform to which the software application is to be installed is initiated using the computing platform icon. Installation of the selected software application is then initiated on the selected computing platform.

The computing platform may be a remote computing platform. In some embodiments, the selection of the software application is received with the use of a graphical user interface in response to the software application icon being pointed to using a pointing device.

In a preferred embodiment of the invention, the pointing device is a mouse and the selection of the computing platform is received in response to the software application icon being dragged over the computing platform icon and the software application icon being dropped. The installation of the selected software application on the selected computing platform is initiated in response to the software application icon being dropped over the computing platform icon.

Within this specification reference to drag and drop has a conventional meaning of selecting an icon and dragging it across the screen to a target icon; notwithstanding, selecting a first icon and then pointing to a target icon, shall have a same meaning and effect throughout this specification. Relatively moving two icons is meant to mean moving one icon toward another.

In some embodiments, upon initialization, the selected computing platform is tested to determine whether it is a valid computing platform.

In some embodiments, upon initialization, a user account is created for the selected software application.

In some embodiments, upon initialization, files specific to the selected application software are installed on the selected computing platform.

In some embodiments, upon initialization, file access permissions are set to allow a user to access the application.

In some embodiments, upon initialization, a console on the selected computing platform is updated with information indicating that the selected software application is resident on the selected computing platform.

In accordance with another broad aspect, the invention provides a method of de-installing a software application from a computing platform comprising the steps of displaying a software application icon representing the software application and a computing platform icon representing the computing platform on which the software application is installed. A selection of the software application for de-installation using the software application icon is received. A selection of the computing platform from which the software application is to be de-installed using the computing platform icon is also received. De-installation of the selected software application from the selected computing platform is then initiated.

The computing platform may be a remote computing platform.

In some embodiments, the displaying comprises displaying the software application as being installed on the computing platform with the use of the software application icon and the computing platform icon.

In some embodiments, the selection of the software application is received with the use of a graphical user interface in response to the software application icon being pointed to using a pointing device.

In some embodiments, the pointing device is a mouse.

In some embodiments, the selection of the computing platform is in response to the software application icon being dragged away from the computing platform icon and the software application icon being dropped.

In some embodiments, upon initialization, the selected computing platform is tested to determine whether it is a valid computing platform for de-installation of the software application.

In some embodiments, upon initialization, any process associated with the selected software application is stopped.

In some embodiments, upon initialization, data file changes specific to the software application are copied back to a storage medium from which the data file changes originated prior to installation.

In some embodiments, upon initialization, a user account associated with the selected software application is removed.

In some embodiments, upon initialization, a console on the computing platform is updated with information indicating that the selected software application is no longer resident on the selected computing platform.

The invention provides a GUI (Graphical User Interface) adapted to perform any of the above method steps for installation or de-installation.

The invention provides a GUI adapted to display a software application icon representative of a software application available for installation and display a computing platform icon representative of a computing platform.

The GUI is responsive to a selection of the software application icon and the computing platform icon by initiating installation of the software application on the computing platform. The invention provides a GUI adapted to display a software application icon representative of a software application and display a computing platform icon representative of a computing platform, the GUI being responsive to a selection of the software application icon

and the computing platform icon by initiating de-installation of the software application from the computing platform.

The GUI is adapted to display a software application icon representative of a software application installed on a computing platform, the GUI being responsive to a selection of the software application icon by initiating de-installation of the software application from the computing platform.

Selection can be made using a drag and drop operation.

The method of de-installation includes displaying a software application icon representing the software application installed on a computing platform. A selection of the software application for de-installation from the computing platform is received using the software application icon. The de-installation of the selected software application from the computing platform is then initiated. In some embodiments, the selection of the application icon is in response to the software application icon being dragged away. In some embodiments, the de-installation of the selected software application from the computing platform is initiated in response to the software application icon being dropped.

Accordingly, the invention relates, in part to methods of installing and removing software applications through a selection such as, for example, a graphical drag and drop operation. In some embodiments of the invention, functions of the GUI support the ability to select an object, move it and initiate an operation when the object is placed on a specific destination. This process has supported operations including the copy and transfer of files. Some embodiments of the invention extend this operation to allow software applications, represented by a graphical icon, to be installed in working order following a drag and drop in a graphical user interface.

The following definitions are used herein:

Storage repository: A centralized disk based storage containing application specific files that make up a software application.

GUI (Graphical User Interface): A computer-based display that presents a graphical interface by which a user interacts with a computing platform by means of icons, windows, menus and a pointing device is referred to as a GUI.

Icon: An icon is an object supported by a GUI. Normally an icon associates an image with an object that can be operated on through a GUI.

Aspects of the invention include:

•GUI supporting drag and drop operations

•Icons

•Storage medium

•Console

•Network connections

•One or more computing platforms

While software applications are represented as graphical icons, they are physically contained in a storage medium. Such a storage medium consists, for example, of disk-based file storage on a server accessible through a network connection. A computing platform is a computer with an installed operating system capable of hosting software applications.

When a software application icon is "dropped" on a computing platform the applications associated with the icon are installed in working order on the selected platform. The computing platform is generally remote with respect to either the platform hosting the graphical interface or the server hosting the storage medium. This topology is not strictly required, but rather a likely scenario.

Referring to Figure 12, shown is a schematic showing the operation of an aspect of the invention, according to an embodiment of the invention.

A graphical icon representing a specific software application is displayed through a graphical user interface. Also displayed is an icon representing a remote computing platform upon which a software application can be hosted. Only one computing platform icon is shown; however, it is to be understood that several computing platform icons each representing a respective remote or local computing platform may also be displayed. Similarly, several software application icons may be displayed depending on the number of software applications available. The software application is selected, through the use of a graphical user interface, by pointing to its software application icon and using a mouse click (or other pointing device). The software application icon is dragged over top of the remote computing platform icon and dropped. The drop initiates the operation of installing software applications on the remote computing platform.

Referring to Figure 13, shown is a schematic diagram illustrating software application icons collected in a centralized file storage medium and a remote computing platform icon, according to another embodiment of the invention. The software applications icons collected in the file storage medium, as shown, are graphical icons each representing respective software applications, which are entries in the file storage medium. The computing platform icon represents a computing platform available to host any one or more of the software applications represented by the software icons. When one of the software application icons is selected, dragged, and dropped on a computing platform icon the respective software applications installed on the remote computing platform corresponding to the computing platform icon.

Referring to Figure 14, a screen snapshot of an implementation is shown according to the schematic of Figure 13. The screen snap shot shows, as an example implementation, software application and computing platform icons. Available software applications corresponding to software application icons ClearCase_Liserver, ClearCase_Registry & Samba2_clone are grouped under the heading "OFFLINE". Computing platforms available to host software applications are featured under the heading "Data Center" and are represented by computing platform icons dell8xx, dell9p, pts2 & pts6.

Operations involve selecting a software application icon, dragging it over a candidate computing platform icon and releasing, or dropping it on the computing platform icon. The selected software application will then be installed in working order on the selected computing platform. The reverse is true for removal of a software application from a selected computing platform. De-installation is accomplished by selecting an application installed on a compute platform and dragging to the repository. The application in question is shown to be associated with a compute platform in graphical fashion. In one embodiment, as shown in Figure 14, applications are associated with a compute platform in hierarchical order, or in a tree view. An application connected to a compute platform as root in a tree view is selected and dropped onto the repository. This initiates the de-install operation.

Referring to Figure 15, shown is a flow chart of a method of initializing the icons of Figure 14. An icon such as a computing platform icon or software application icon is created as a GUI (Graphical User Interface) object. Drop handler operations are then associated to the computing platform icon. In particular, any operation required for installation is associated with the icon.

With reference to Figures 12 to 14, the installation process is initiated by a drag and drop of a software application icon, from a storage medium, to a top of a computing platform icon. An install drop handler implements the installation of the software application. The install drop handler performs several tasks on the destination computing platform, which:

•Tests whether the computing platform is a valid computing platform;
•Creates a user account for the software application;
•Installs application specific files so that they are accessible to the remote computing platform;
•Sets file access permissions such that a new user can access its applications;
•Starts a first application; and
•Updates a console showing that the selected software application is resident on the selected computing platform.

These steps will now be described with reference to Figure 16 which is a flow chart of a method of installing a software application on a computing platform in response to the drag and drop operation of Figure 2. As discussed above, in operation the installation process is invoked when a software application icon is dropped on a computing platform icon.

The method steps of Figure 16 are performed by an install drop handler. During installation, verification is performed to determine whether the selected computing platform is a valid candidate for installing a selected software application. If the computing platform is a valid candidate, a user account is created for the software application; otherwise, the installation process ends. Once the user account is created, application files associated with the software applications are installed on the selected computing platform so that they are accessible to the computing platform. File access permissions are then set such that one or more new users can access the application being installed. A first application is then started. In some embodiments, an application, for example accounting or payroll represent several programs/processes. In order to start the accounting/payroll service a first application is started that may in turn start other programs/processes. The first program is started. It is then up to the specific service, accounting/payroll, to start any other services it requires.

A console on the selected computing platform on which the software application is being installed is then updated with information to show that the selected software application is resident on the selected computing platform. The console is operational on any desktop computing platform for example, Unix, Linux and Windows.

With reference to Figure 17, the removal process is initiated by a drag and drop operation of a software application icon from a computing platform icon to the repository. For this purpose each computing platform icon shows, with the use of associated software application icons (not shown in Figure 15), each application software installed on the computing platform.

The de-installation of application software from a selected computing platform is done by a remove drop handler which performs the following tasks on the selected computing platform:

•Tests whether the computing platform is a valid computing platform; Tests are made to verify whether the computing platform is operational. For example, it may have been taken off-line due to a problem or routine maintenance. If so, then applications should not be removed or installed until this state changes.

•Stops processes associated with the software application;
•Copies application specific data file changes from the computing platform back to the storage medium;
•Removes user accounts associated with the software application; and
•Updates the console to show that the selected software application is resident in the repository.

These steps will now be described with reference to Figure 17 which is a flow chart of a method of de-installing a software application from a computing platform in response to a drag and drop operation of Figure 13. The state of the platform is verified to determine whether it is valid. The state includes, for example, on-line, off-line (a problem state) or maintenance (off-line, but for a scheduled task). If the state of the platform is valid and a process or processes associated with a software application selected to be de-installed are stopped; otherwise the de-installation process ends. Once the processes are stopped, application specific data file changes are copied from the computing platform back to the storage medium. This is a process of capturing changes that may have taken place while the application ran and that need to be saved for future instances when the application runs again. For example, payroll may be run every other week. Changes made to the system, accrued amounts, year to date payments, etc. will need to be saved so that when payroll is run the next time these values are updated. Depending on how the operator configures a system, it may be required to copy changes made when payroll ran back to the repository so that the next time payroll is run these values are installed along with the application.

Any user account associated with the selected software application is removed and a console on the computing platform from which the selected software application is being de-installed is updated with information to show that the selected software application is resident in the repository. Using the method steps of Figures 16 and 17, software applications are therefore installed on a computing platform and removed from the computing platform through a graphical user interface drag and drop operation.

A number of programming languages may be used to implement programs used in embodiments of the invention. Some example programming languages used in embodiments of the invention include, but are not limited to, C++, Java and a number of scripting languages including TCL/TK and PERL.

Installation of software applications is preferably performed on computing platforms that are remote from a console computing platform. However, some embodiments of the invention also have installation of software applications performed on a computing platform that is local to a console computing platform. Numerous modifications and variations of the present invention are possible in light of the above teachings. It is therefore to be understood that within the scope of the appended claims, the invention may be practiced otherwise than as specifically described herein.

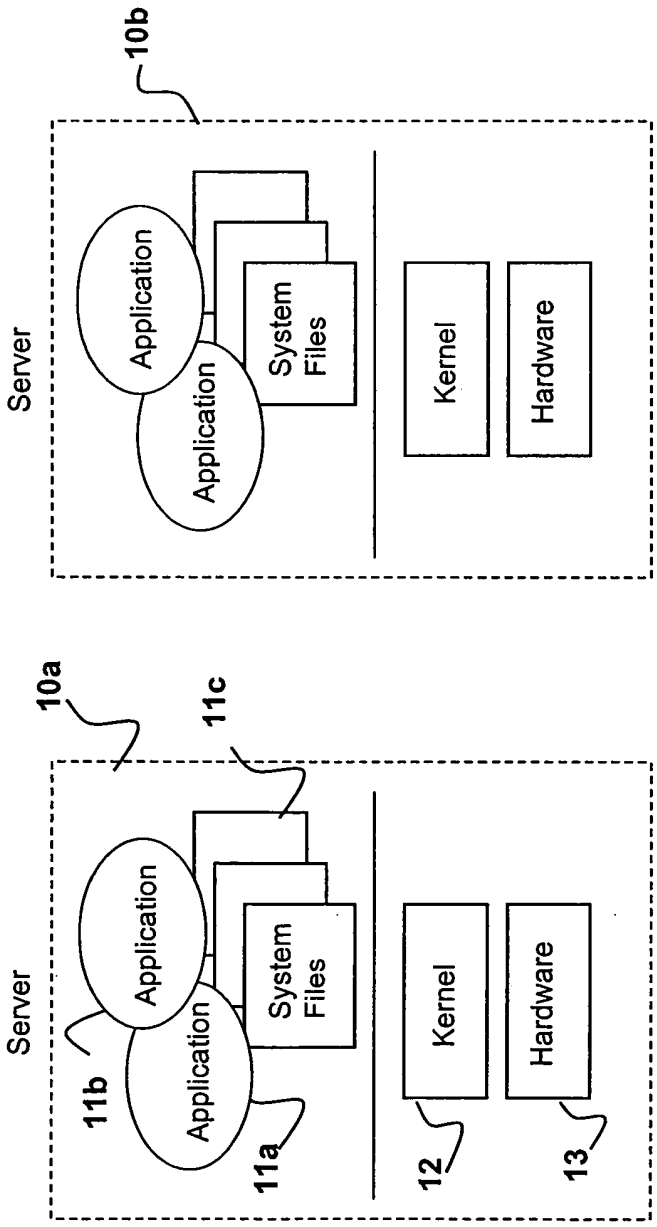


Figure 1

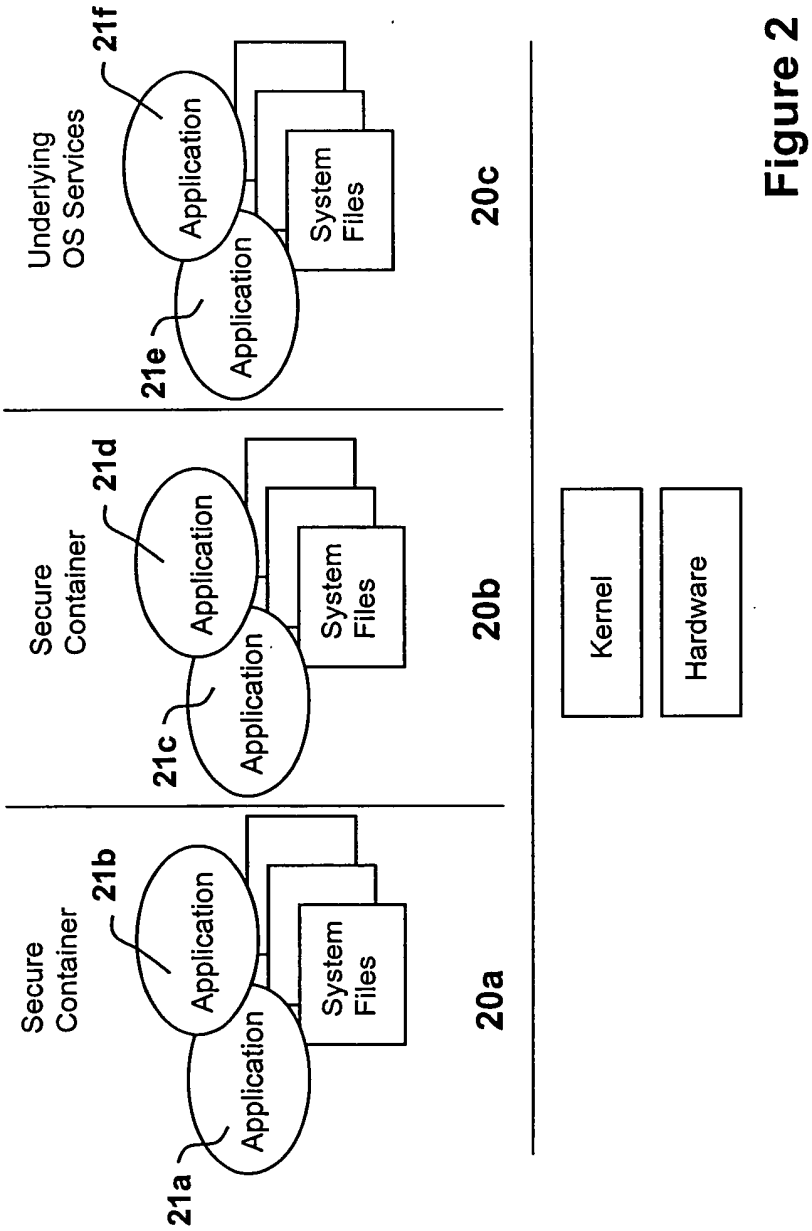

Underlying OS Services
21f
Application
Application
System Files
21e
20c
Secure Container
21d
Application
Application
System Files
21c
20b
Secure Container
21b
Application
Application
System Files
21a
20a
Kernel
Hardware

Figure 2

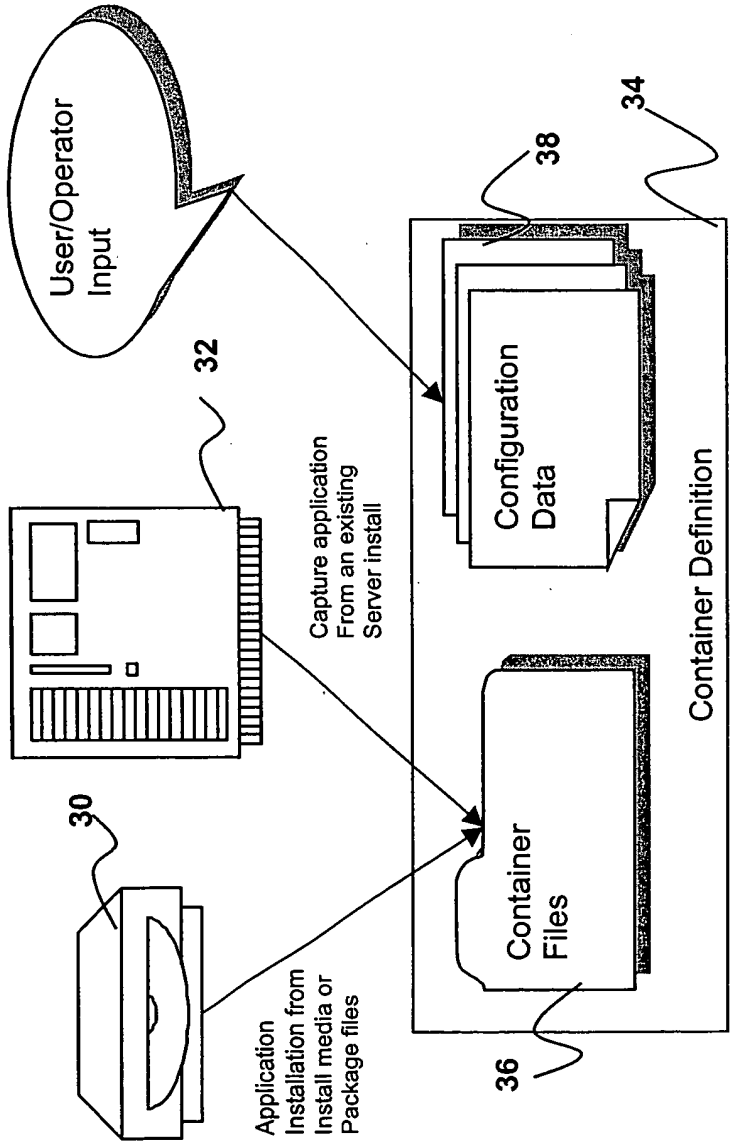

User/Operator Input
32
Capture application From an existing Server install
38
34
Configuration Data
Container Definition
30
Container Files
Application Installation from Install media or Package files
36

Figure 3

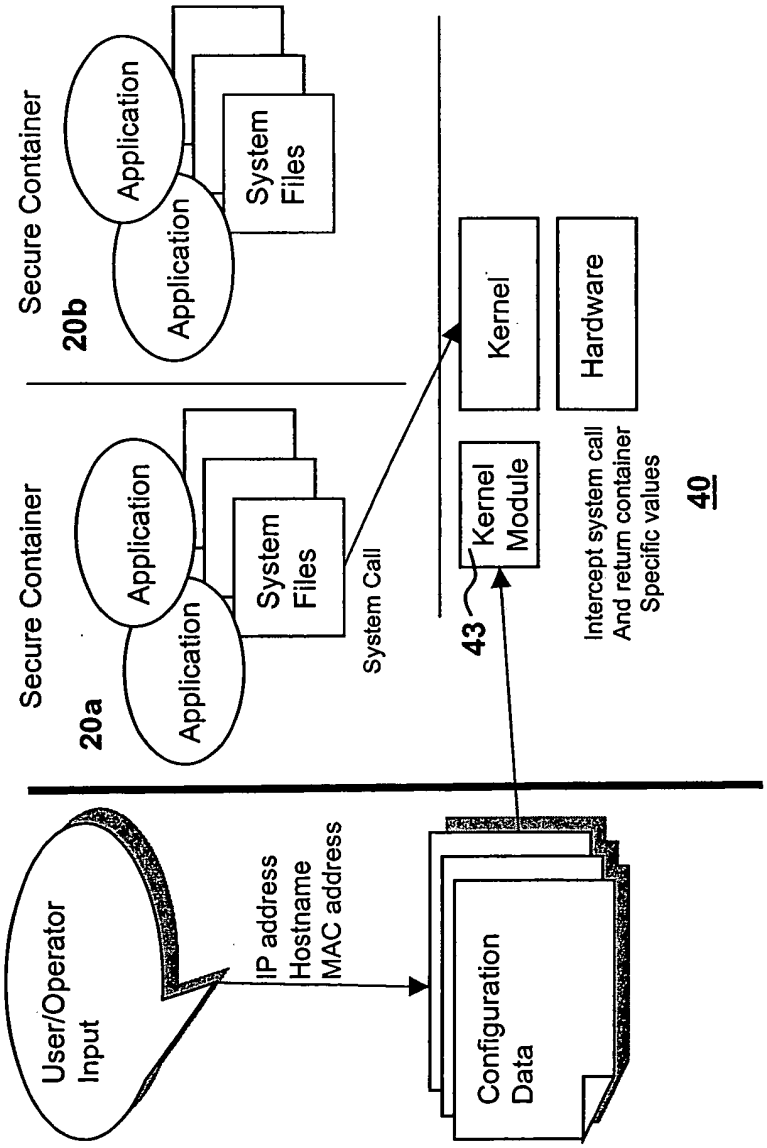

Secure Container
20b
Application
Application
System Files
Secure Container
20a
Application
Application
System Files
System Call
Kernel
Kernel Module
43
Hardware
Intercept system call
And return container
Specific values
40
User/Operator Input
IP address
Hostname
MAC address
Configuration Data

Figure 4

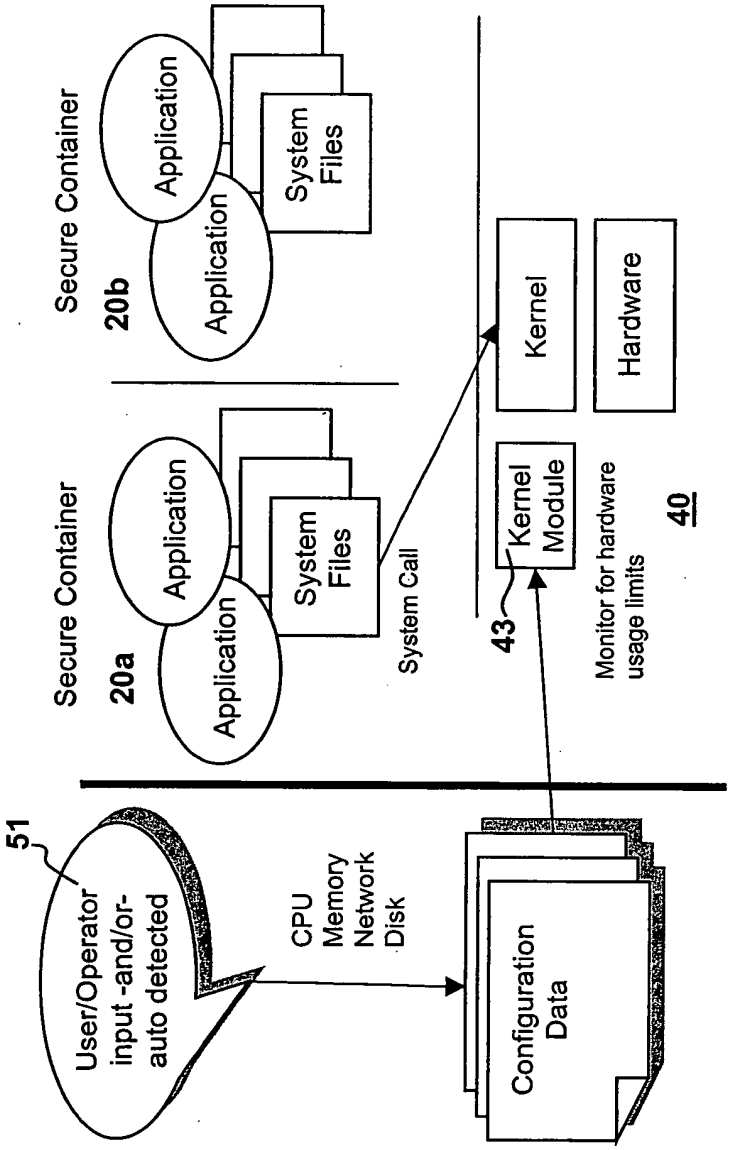

Secure Container
20b
Application
Application
System Files
Secure Container
20a
Application
Application
System Files
System Call
Kernel
Hardware
Kernel Module
43
Monitor for hardware usage limits
40
51
User/Operator input -and/or- auto detected
CPU
Memory
Network
Disk
Configuration Data

Figure 5

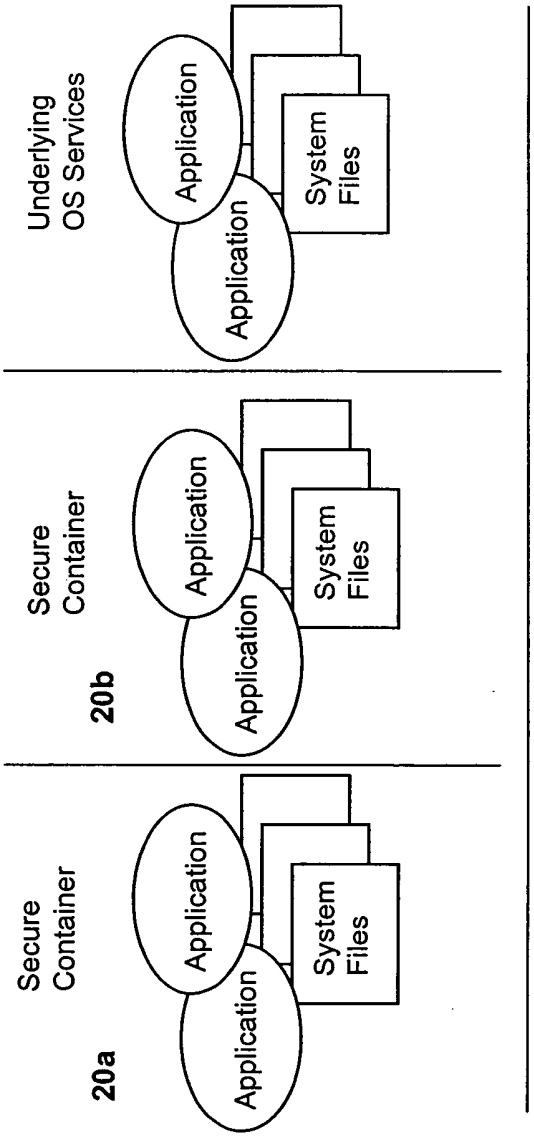

Underlying OS Services
Application
Application
System Files
Secure Container
20b
Application
Application
System Files
Secure Container
20a
Application
Application
System Files
Kernel
Hardware
40

Figure 6

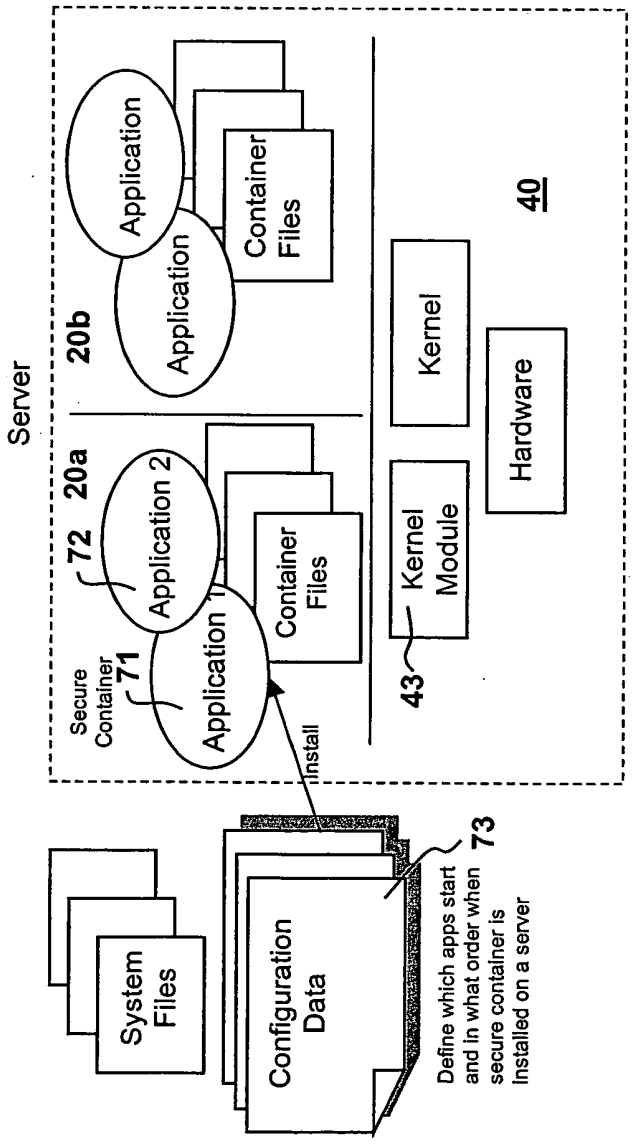

Server
20b
Application
Application
Container Files
Kernel
Hardware
40
20a
72
Application 2
Container Files
Secure Container
71
Application 1
Kernel Module
43
Install
System Files
Configuration Data
73
Define which apps start and in what order when secure container is installed on a server

Figure 7

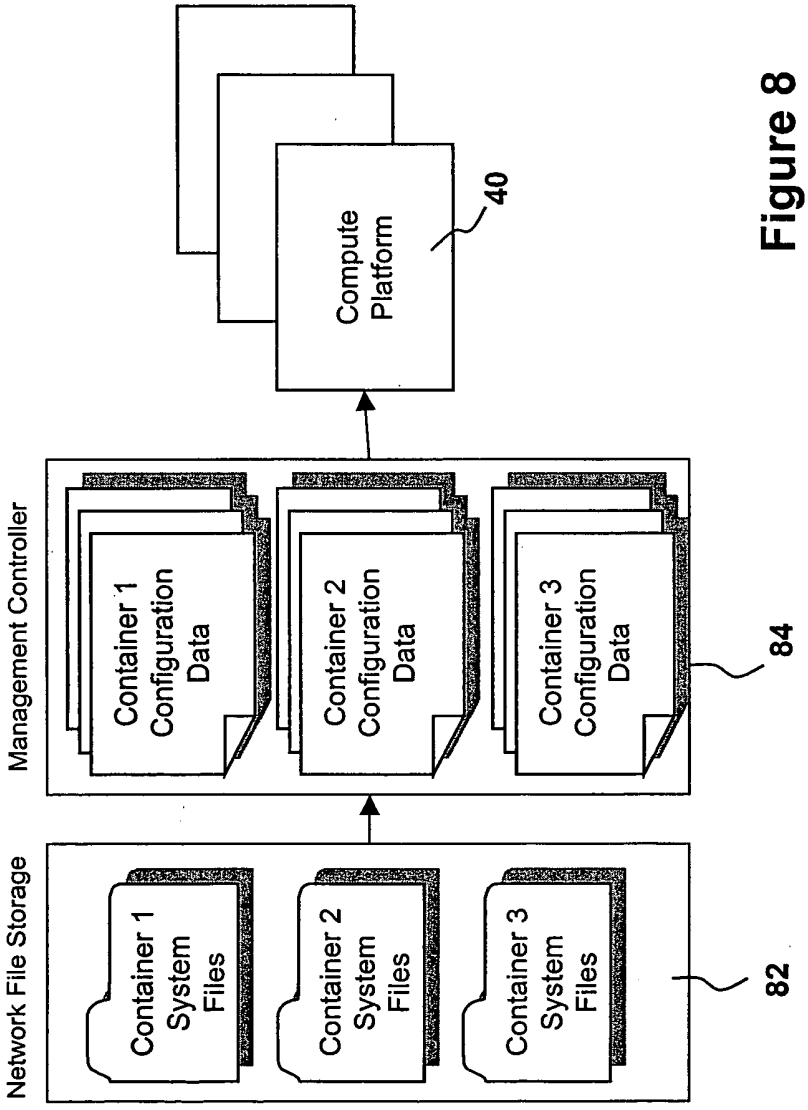

Compute Platform
40
Management Controller
Container 1 Configuration Data
Container 2 Configuration Data
Container 3 Configuration Data
84
Network File Storage
Container 1 System Files
Container 2 System Files
Container 3 System Files
82

Figure 8

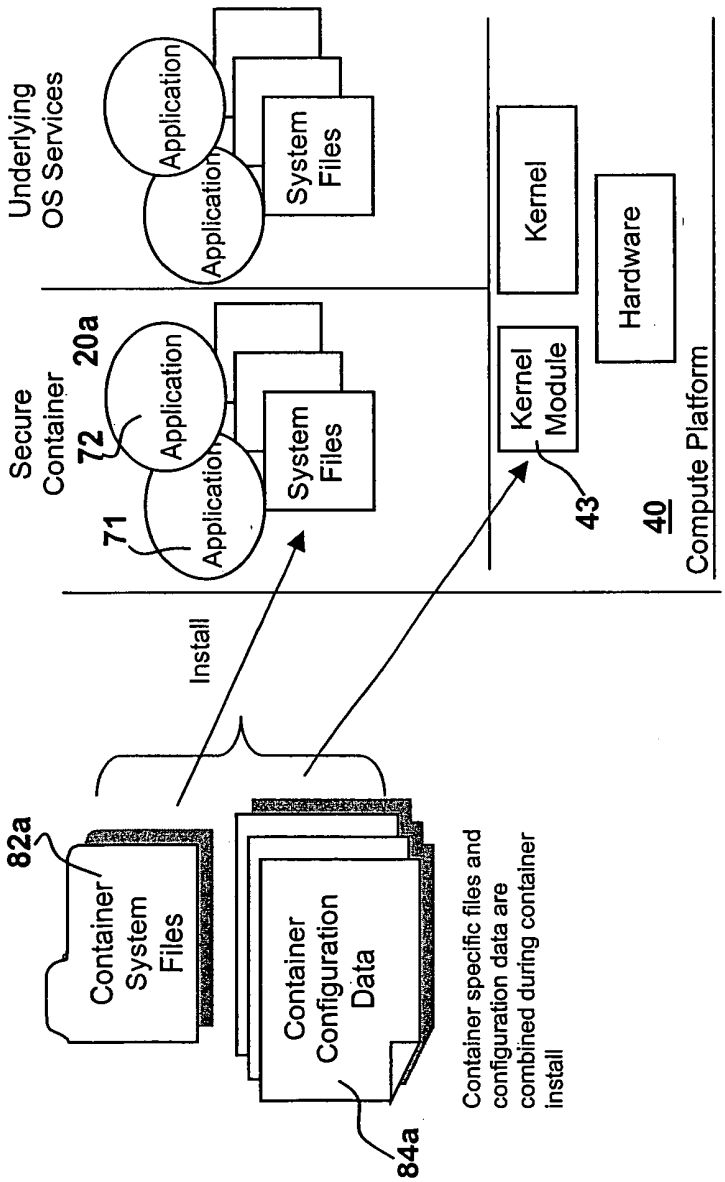

Underlying OS Services
Application
Application
System Files
Secure Container
20a
72
Application
Application
System Files
71
Kernel
Kernel Module
Hardware
43
40
Compute Platform
Install
82a
Container System Files
Container Configuration Data
84a
Container specific files and configuration data are combined during container install

Figure 9

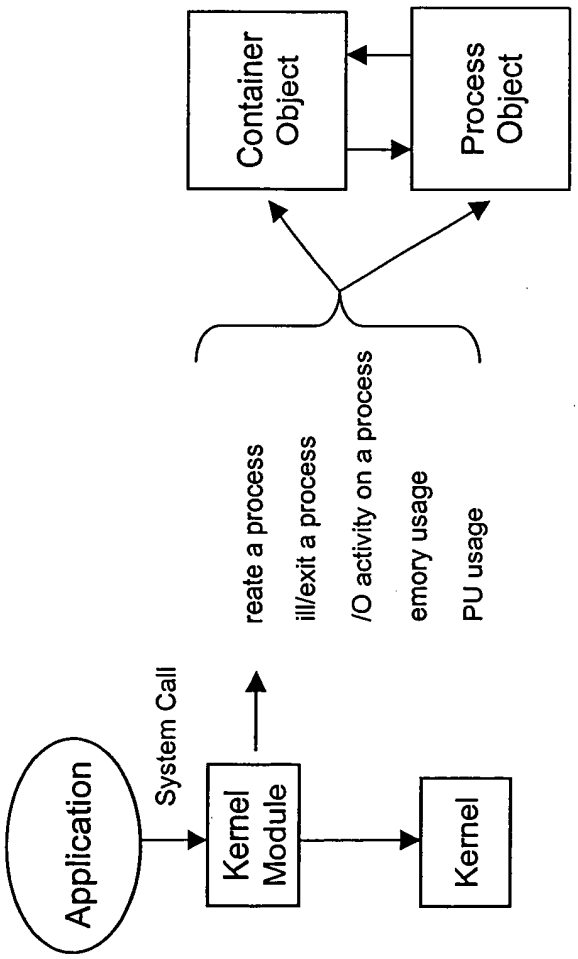

Container Object
Process Object
reate a process
ill/exit a process
/O activity on a process
emory usage
PU usage
System Call
Application
Kernel Module
Kernel

Figure 10

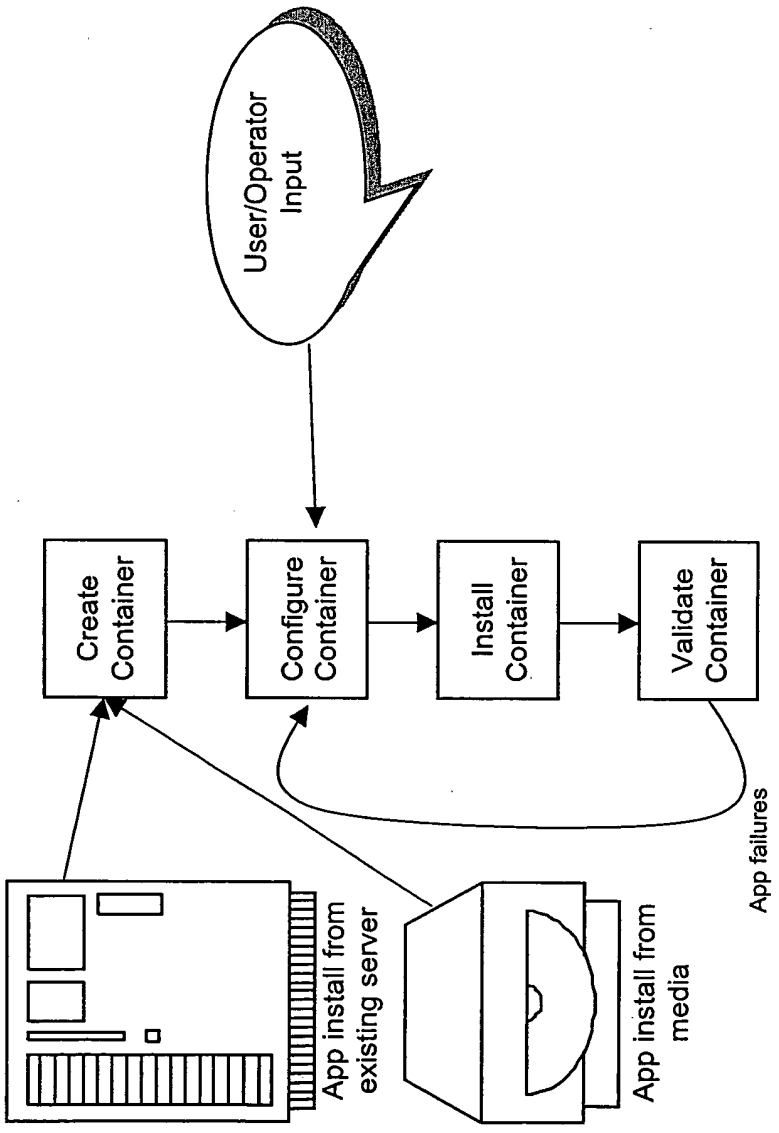

User/Operator Input
Create Container
Configure Container
Install Container
Validate Container
App failures
App install from existing server
App install from media

Figure 11

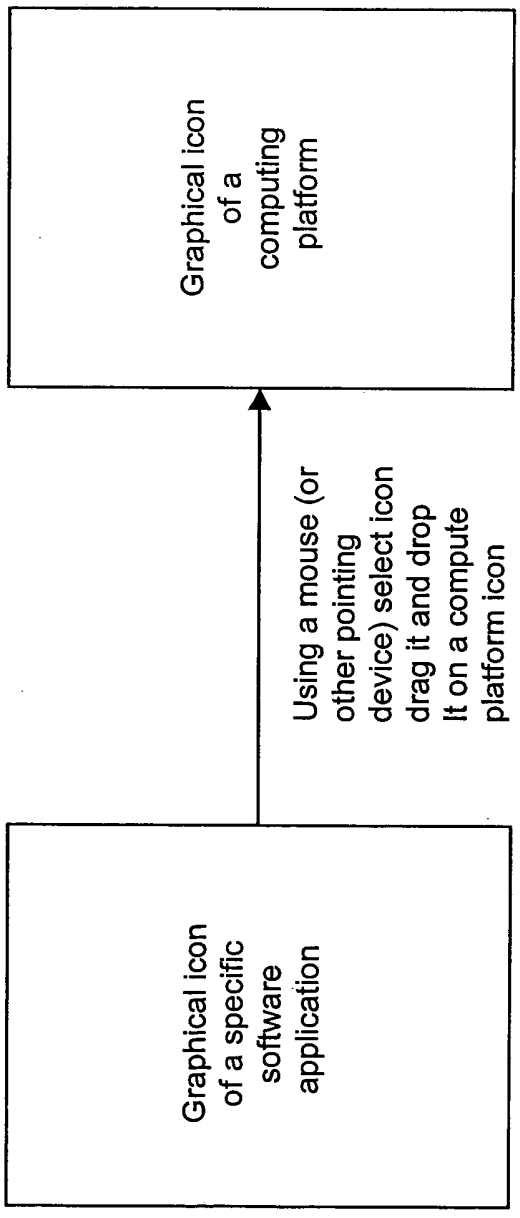

Graphical icon of a computing platform
Using a mouse (or other pointing device) select icon drag it and drop It on a compute platform icon
Graphical icon of a specific software application

Figure 12

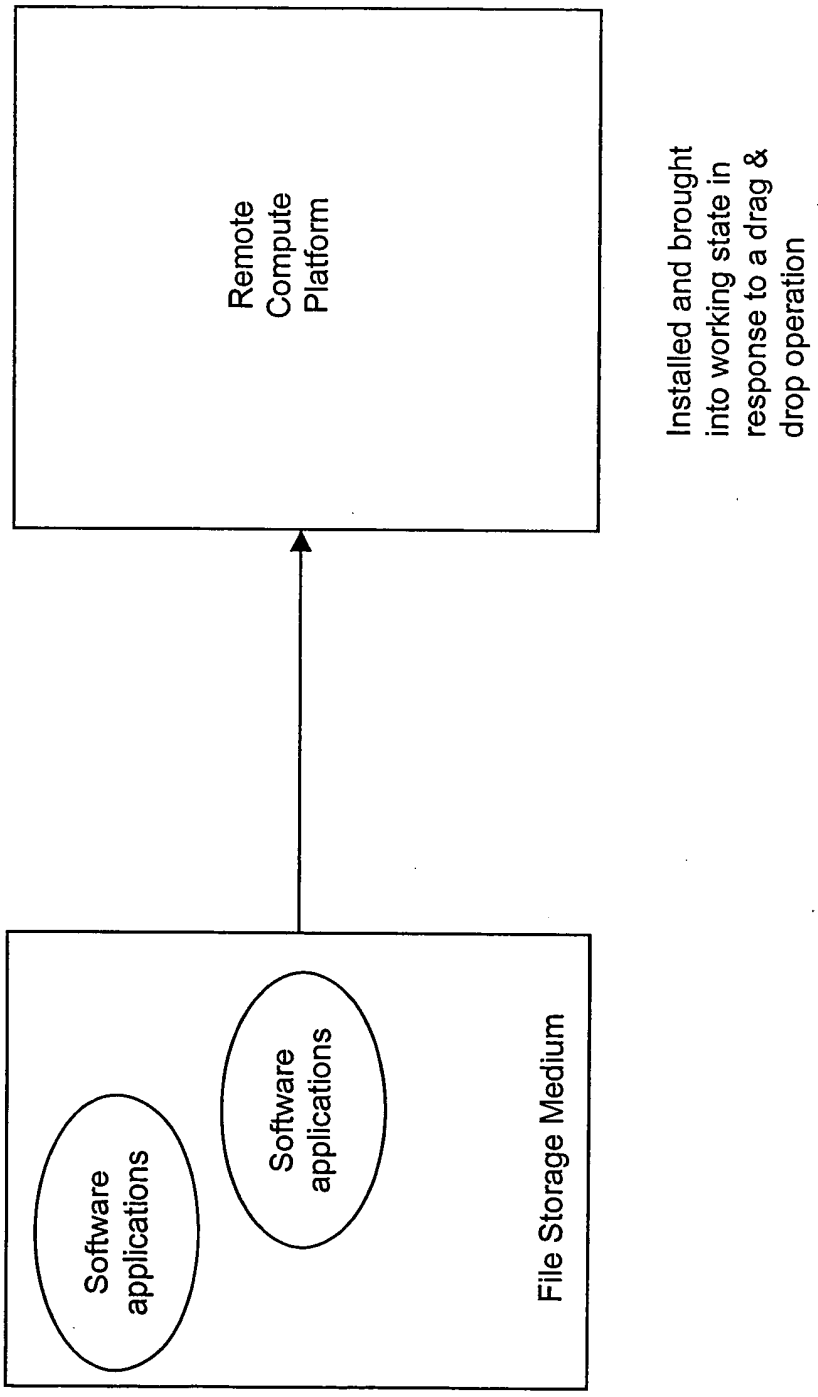

Remote Compute Platform
Installed and brought into working state in response to a drag & drop operation
Software applications
Software applications
File Storage Medium

Figure 13

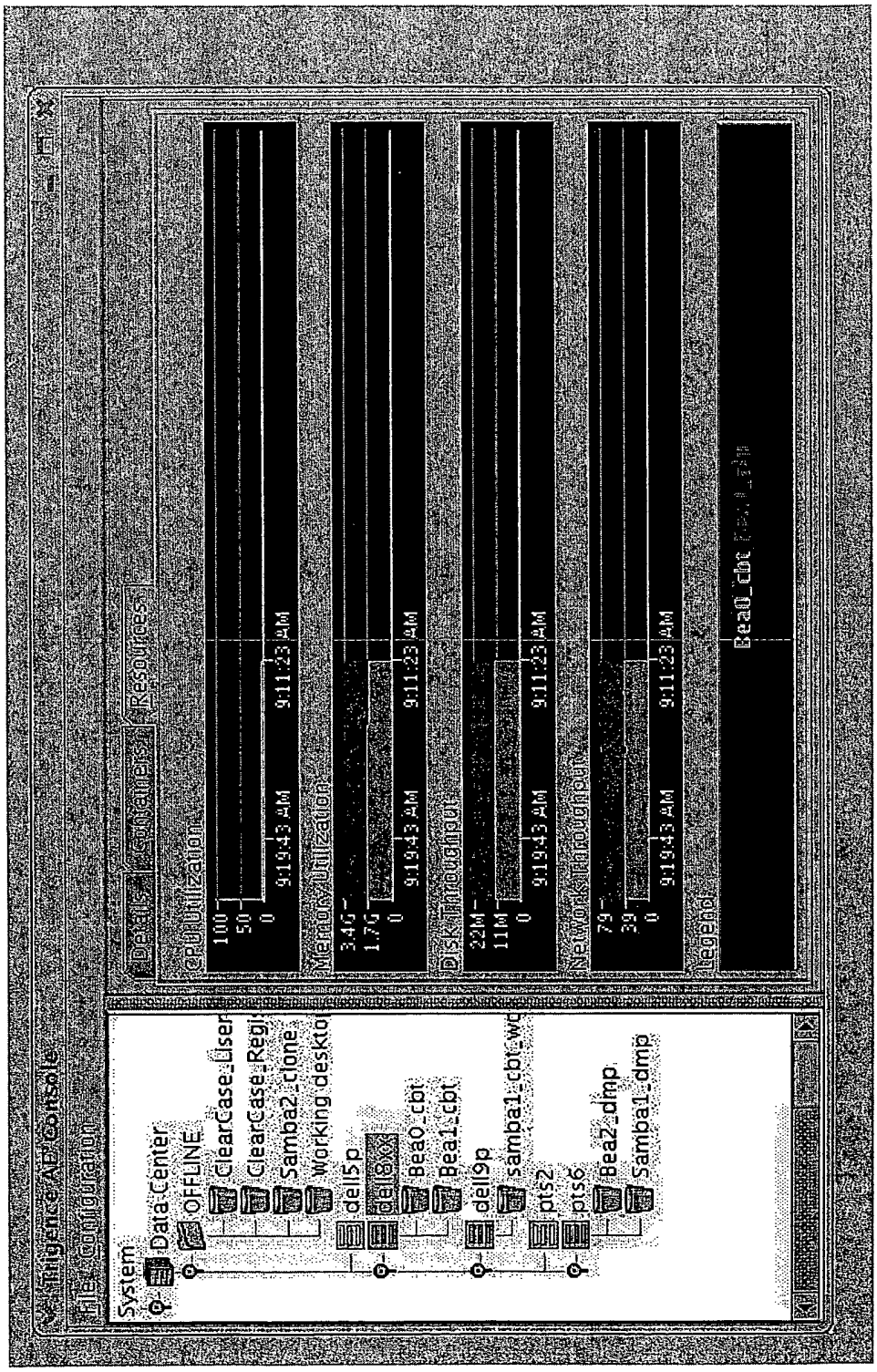

VERITAS Intelligence API Console
File Configuration
System
Data Center
OFFLINE
ClearCase_User
ClearCase_Regi
Samba2_clone
working_desktol
dell5p
dell6xx
Bea0_cbt
Bea1_cbt
dell9p
samba1_cbt_wo
pts2
pts6
Bea2_dmp
Samba1_dmp
Details
Sequences
Resources
CPU Utilization
100
50
0
9:19:43 AM
9:11:23 AM
Memory Utilization
3.4G
1.7G
0
9:19:43 AM
9:11:23 AM
Disk Throughput
22M
11M
0
9:19:43 AM
9:11:23 AM
Network Throughput
79
39
0
9:19:43 AM
9:11:23 AM
Legend
Bea0_cbt

Figure 14

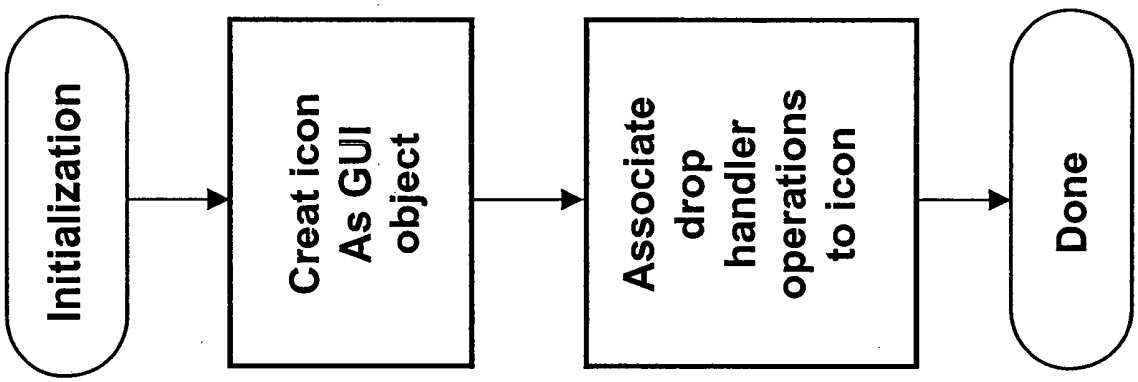

Initialization
Creat icon As GUI object
Associate drop handler operations to icon
Done

Figure 15

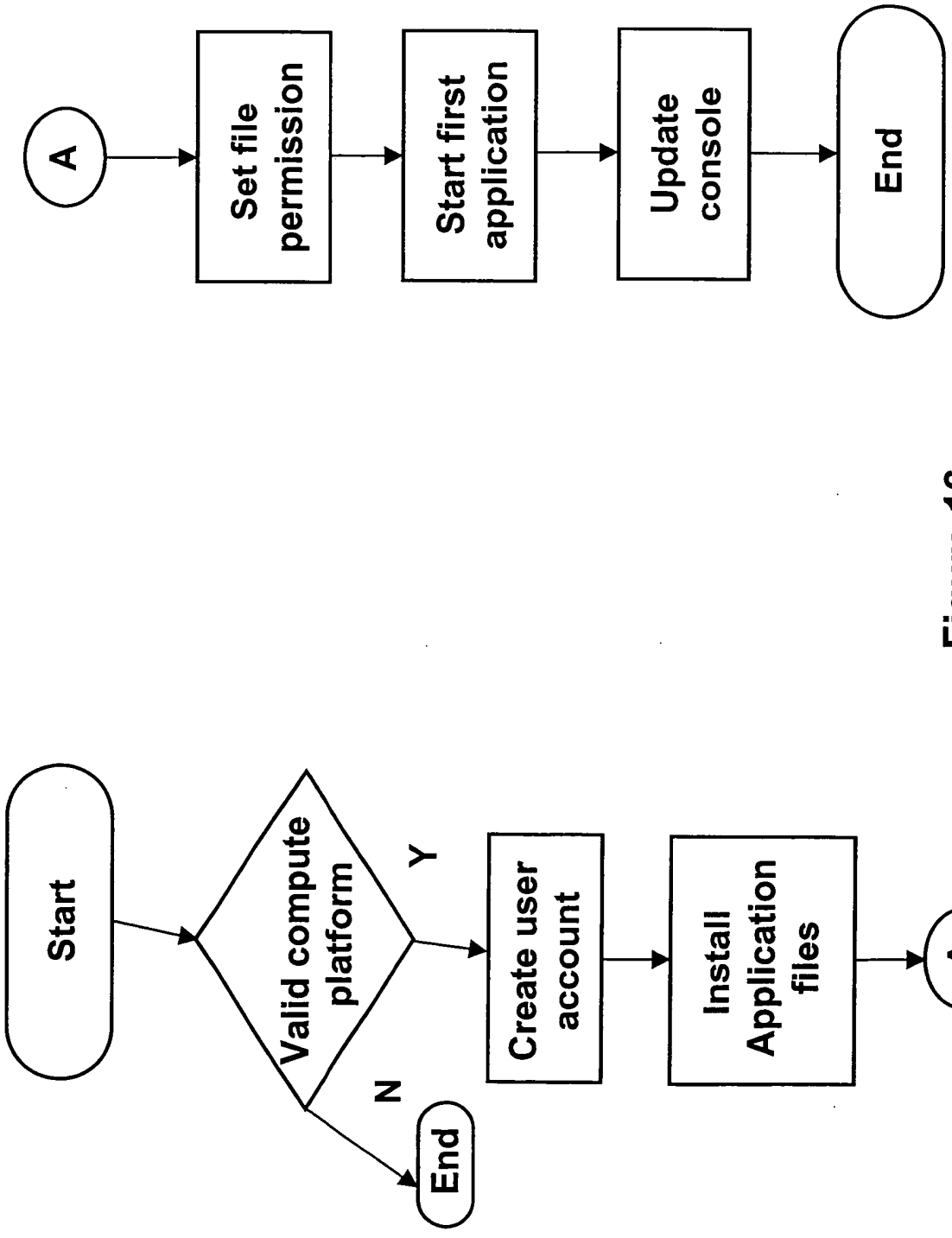

Start
Valid compute platform
Y
N
End
Create user account
Install Application files
A
A
Set file permission
Start first application
Update console
End

Figure 16

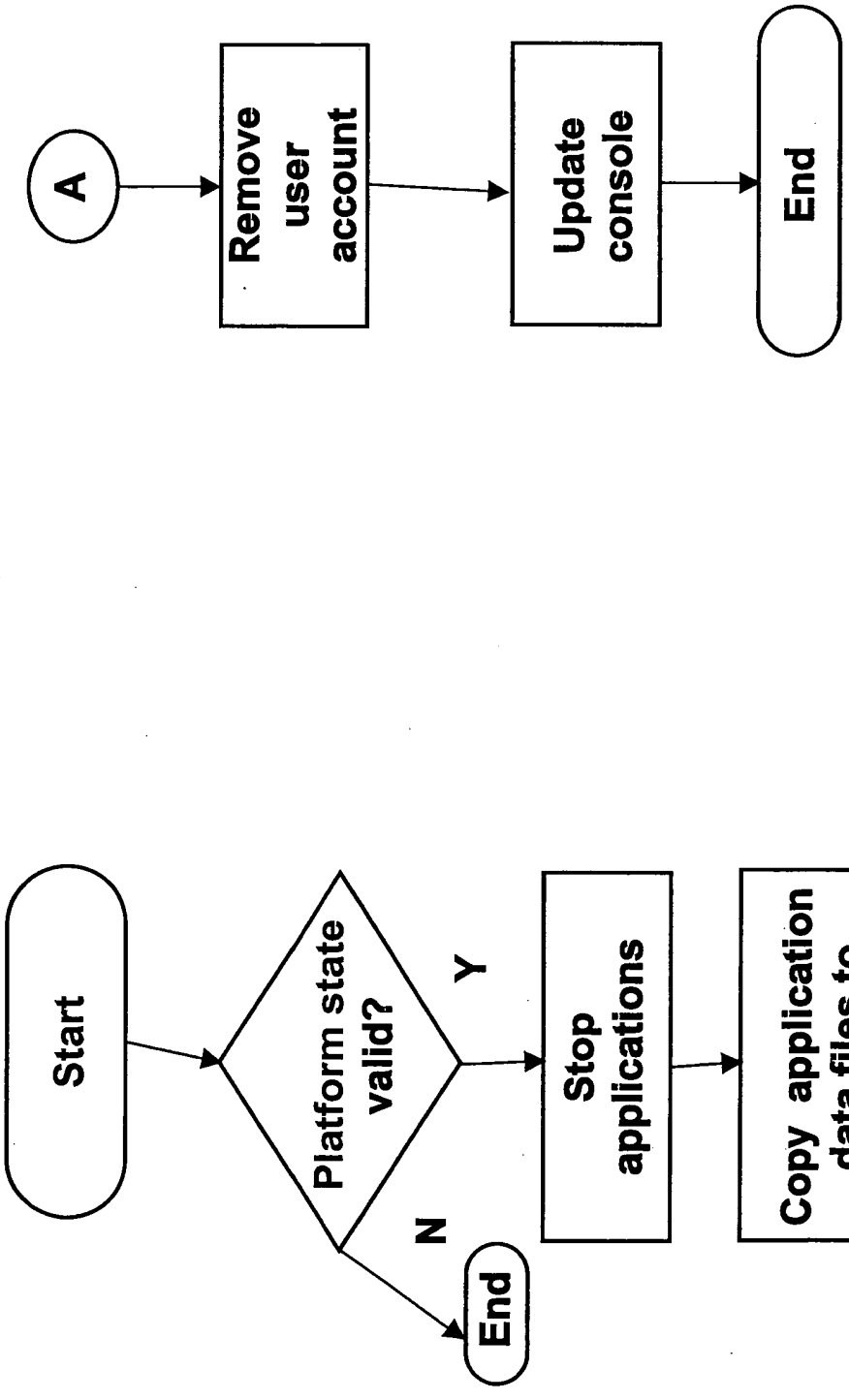

Start
Platform state valid?
Y
N
End
Stop applications
Copy application data files to repository
A
A
Remove user account
Update console
End

Figure 17

Antrag auf Erteilung eines europäischen Patents / Request for grant of a European patent / Requête en délivrance d'un brevet européen

1

Bestätigung einer bereits durch Telefax eingereichten Anmeldung / Confirmation of an application already filed by facsimile / Confirmation d'une demande déjà déposée par téléfax
Wenn ja, Datum der Übermittlung des Telefax und Name der Einreichungsbehörde / If yes, facsimile date and name of the authority with which the documents were filed / Si oui, date d'envoi du téléfax et nom de l'autorité de dépôt

☐ Ja / Yes / Oui

Datum / Date | Behörde / Authority / Autorité

Nur für amtlichen Gebrauch / For official use only / Cadre réservé à l'administration

Anmeldenummer / Application No. / N° de la demande	MKEY	1	04021916.4
Tag des Eingangs (Regel 24(2)) / Date of receipt (Rule 24(2)) / Date de réception (règle 24(2))	DREC	2	15.09.04
Tag des Eingangs beim EPA (Regel 24(4)) / Date of receipt at EPO (Rule 24(4)) / Date de réception à l'OEB (règle 24(4))	RENA	3	
Anmeldetag / Date of filing / Date de dépôt		4	

Tabulatoren-Positionen / Tabulation marks / Arrêts de tabulation

Es wird die Erteilung eines europäischen Patents und gemäß Artikel 94 die Prüfung der Anmeldung beantragt / Grant of a European patent, and examination of the application under Article 94, are hereby requested / Il est demandé la délivrance d'un brevet européen et, conformément à l'article 94, l'examen de la demande — EXAM 4 — 5

☒ Prüfungsantrag in einer zugelassenen Nichtamtssprache (siehe Merkblatt II, 5): / Request for examination in an admissible non-EPO language (see Notes II,5): / Requête en examen dans une langue non officielle autorisée (voir notice II,5) :

Zeichen des Anmelders oder Vertreters (max. 15 Positionen) / Applicant's or representative's reference (maximum 15 spaces) / Référence du demandeur ou du mandataire (max. 15 caractères ou espaces) — AREF — 6: 14738EP /nh

Anmelder / Applicant / Demandeur

Name / Nom — 7: Trigence Corp.

Anschrift / Address / Adresse — 8:

APPR 01 # 498RR108

DEST

750 Palladium Drive, Ste 210
Ottawa, Ontario
Canada K2V 1C7

Zustellanschrift / Address for correspondence / Adresse pour la correspondance — 9:

PADR

Staat des Wohnsitzes oder Sitzes / State of residence or of principal place of business / Etat du domicile ou du siège — 10: CA

Staatsangehörigkeit / Nationality / Nationalité — 11: CA

Telefon / Telephone / Téléphone — 12:

Telex / Télex | Telefax / Fax / Télécopie — 13:

Weitere(r) Anmelder auf Zusatzblatt / Additional applicant(s) on additional sheet / Autre(s) demandeur(s) sur feuille supplémentaire — 14: ☐

Vertreter / Representative / Mandataire

Name / Nom — 15: BANZER, Hans-Jörg, Dipl.-Ing., et al

(Nur einen Vertreter angeben, der in das europäische Patentregister einzutragen ist und an den zugestellt wird / Name **only one** representative who is to be listed in the Register of European Patents and to whom notification is to be made / N'indiquer qu'**un seul** mandataire, qui sera inscrit au Registre européen des brevets et auquel signification sera faite)

FREP 01 83617 48 # # #

Geschäftsanschrift / Address of place of business / Adresse professionnelle — 16:

KRAUS & WEISERT
Thomas-Wimmer-Ring 15
80539 München
Deutschland

Telefon / Telephone / Téléphone — 17: 089-290-60-0

Telex / Télex | Telefax / Fax / Télécopie — 18: 089-290-60-111

Weitere(r) Vertreter auf Zusatzblatt / Additional representative(s) on additional sheet / Autre(s) mandataire(s) sur feuille supplémentaire — 19: ☐

TRAN FILL EW

EPA/EPO/OEB Form 1001.1 05.04

14738EP /nh

Raum für Zeichen des Anmelders / Space for applicant's reference / Espace réservé à la référence du demandeur

Vollmacht / Authorisation / Pouvoir

ist beigefügt / is enclosed / joint — 20 ☐

ist registriert unter Nummer / has been registered under No. / a été enregistré sous le n° GENA — 21 ☐ Nummer / Number / Numéro

Erfinder / Inventor / Inventeur INVT 20 # #

Anmelder ist (sind) alleinige(r) Erfinder / The applicant(s) is (are) the sole inventor(s) / Le(s) demandeur(s) est (sont) le (les) seul(s) inventeur(s) — 22 ☐

Erfindernennung in gesondertem Schriftstück / Designation of inventor attached / Voir la désignation de l'inventeur ci-jointe — 23 ☒

Bezeichnung der Erfindung / Title of invention / Titre de l'invention — 24

TIDE TIEN TIFR

SYSTEM FOR CONTAINERIZATION OF APPLICATION SETS

Prioritätserklärung / Declaration of priority / Déclaration de priorité PRIO — 25

	Staat / State / Etat	Anmeldetag / Date of filing / Date de dépôt	Aktenzeichen / Application No. / N° de la demande
01 # . . # . . . . . #	1 US	Sept. 15, 2003	60/502,619
02 # . . # . . . . . #	2 US	Oct. 20, 2003	60/512,103
03 # . . # . . . . . #	3		
04 # . . # . . . . . #	4		

Weitere Prioritätserklärung(en) auf Zusatzblatt / Additional declaration(s) of priority on additional sheet / Autre(s) déclaration(s) de priorité sur feuille supplémentaire ☐

Es wird hiermit erklärt, daß die Anmeldung eine vollständige Übersetzung der früheren Anmeldung ist (Regel 38(5)) / It is hereby declared that the application is a complete translation of the previous application (Rule 38(5)) / Il est déclaré par la présente que la demande est une traduction intégrale de la demande antérieure (règle 38(5)) PRIO 6 — 25a ☐

Biologisches Material

Die Erfindung bezieht sich auf bzw. verwendet biologisches Material, das nach Regel 28 hinterlegt worden ist.

Biological material

The invention relates to and/or uses biological material deposited under Rule 28.

Matière biologique

L'invention concerne et/ou utilise de la matière biologique, déposée conformément à la règle 28. — 26 ☐

Die Angaben nach Regel 28(1)c) (falls noch nicht bekannt, die Hinterlegungsstelle und das (die) Bezugszeichen [Nummer, Symbole usw.] des Hinterlegers) sind in den technischen Anmeldungsunterlagen enthalten auf / The particulars referred to in Rule 28(1)(c) (if not yet known, the depository institution and the identification reference(s) [number, symbols, etc.] of the depositor) are given in the technical documents in the application on / Les indications visées à la règle 28(1)c) (si pas encore connues, l'autorité de dépôt et la (les) référence(s) d'identification [numéro ou symboles etc.] du déposant) figurent dans les pièces techniques de la demande à la /aux

BIOM 1 # | | | | | | #

Seite(n) / page(s)	Zeile(n) / line(s) / ligne(s)

27

werden später mitgeteilt / will be submitted later / seront communiquées ultérieurement — 27a ☐

Die Empfangsbescheinigung(en) der Hinterlegungsstelle ist (sind) beigefügt / The receipt(s) of deposit issued by the depositary institution is (are) enclosed / Le(s) récépissé(s) de dépôt délivré(s) par l'autorité de dépôt est (sont) joint(s) — 27b ☐

wird (werden) nachgereicht / will be filed later / sera (seront) produit(s) ultérieurement — 27c ☐

14738EP /nh

Raum für Zeichen des Anmelders / Space for applicant's reference / Espace réservé à la référence du demandeur

Falls das biologische Material nicht vom Anmelder, sondern von einem Dritten hinterlegt wurde: / Where the biological material has been deposited by a person other than the applicant: / Lorsque la matière biologique a été déposée par une personne autre que le demandeur:

Ermächtigung nach Regel 28(1)d) / Authorisation under Rule 28(1)(d) / L'autorisation en vertu de la règle 28(1)d)

ist beigefügt / is enclosed / est jointe — 28a ☐

wird nachgereicht / will be filed later / sera produite ultérieurement — 28b ☐

28 Name und Anschrift des Hinterlegers / Name and address of depositor / Nom et adresse du déposant :

Verzicht auf die Verpflichtung des Antragstellers nach Regel 28(3) in gesondertem Schriftstück / Waiver of the right to an undertaking from the requester pursuant to Rule 28(3) attached

29 ☐ Renonciation, sur document distinct, à l'engagement du requérant au titre de la règle 28(3)

Gemäß Regel 28(4) wird hiermit mitgeteilt, daß der Zugang zu dem in den Feldern 26 und 27 genannten biologischen Material nur durch Herausgabe einer Probe an einen Sachverständigen hergestellt wird / It is hereby declared under Rule 28(4) that the availability of the biological material referred to in Sections 26 and 27 shall be effected only by the issue of a sample to an expert

BIOM 3

30 ☐ Conformément à la règle 28(4), il est déclaré par la présente que l'accessibilité à la matière biologique mentionée aux rubriques 26 et 27 ne peut réalisée que par la remise d'un échantillon à un expert

Nucleotid- und Aminosäuresequenzen / Nucleotide and amino acid sequences / Séquences de nucléotides et d'acides aminés

SEQL 1

31

Die Beschreibung enthält ein Sequenzprotokoll nach Regel 27a(1) / The description contains a sequence listing in accordance with Rule 27a(1) ☐ La description contient une liste de séquences selon la règle 27bis(1)

Der vorgeschriebene Datenträger ist beigefügt / The prescribed data carrier is enclosed ☐ Le support de données prescrit est joint

Es wird hiermit erklärt, daß die auf dem Datenträger gespeicherte Information mit dem schriftlichen Sequenzprotokoll übereinstimmt (Regel 27a(2)) / It is hereby stated that the information recorded on the data carrier is identical to the written sequence listing (Rule 27a(2)) ☐ Il est déclaré par la présente que l'information figurant sur le support de données est identique à celle que contient la liste de séquences écrite (règle 27bis(2))

Benennung der Vertragsstaaten und Erklärungen hierzu

Designation of contracting states and associated declarations

DEST

32

Désignation d'Etats contractants et déclarations à ce propos

1. Hiermit werden sämtliche Vertragsstaaten des EPÜ benannt, die diesem bei Einreichung dieser Anmeldung angehören*.

1. All states which are contracting states to the EPC at the filing of this application are hereby designated*.

☒ 1. Sont désignés tous les Etats qui sont des Etats parties à la CBE à la date du dépôt de la présente demande*.

2a. Es ist derzeit beabsichtigt, den **siebenfachen** Betrag einer Benennungsgebühr zu entrichten. Damit gelten die Benennungsgebühren für alle Vertragsstaaten als entrichtet (Art. 2 Nr. 3 GebO).

2a. It is currently intended to pay **seven times** the amount of the designation fee. The designation fees for all the contracting states are thereby deemed to have been paid (Art. 2, No. 3, RFees).

☒ 2a. Il est actuellement envisagé de payer un montant correspondant à **sept fois** la taxe de désignation. Les taxes de désignation sont ainsi réputées payées pour tous les Etats contractants (art. 2, point 3 du RRT).

2b. Abweichend von der Erklärung in Nr. 2a ist derzeit beabsichtigt, **weniger als sieben** Benennungsgebühren für folgende Vertragsstaaten zu entrichten *(bitte Ländercodes und Vertragsstaaten angeben*):*

2b. The declaration in No. 2a does not apply. Instead, it is currently intended to pay **fewer than seven** designation fees for the following contracting states *(please indicate country codes and contracting states*):*

☐ 2b. Contrairement à ce qui est indiqué au n° 2a, il est actuellement envisagé de payer **moins de sept** taxes de désignation pour les Etats contractants suivants *(prière d'indiquer codes de pays et Etats contractants*) :*

(1) ____ (2) ____ (3) ____ (4) ____ (5) ____ (6) ____

Es wird beantragt, für die unter Nr. 2b nicht aufgeführten Vertragsstaaten von der Zustellung von Mitteilungen nach Regel 85a(1) und Regel 69(1) abzusehen.

No communications under Rules 85a(1) or 69(1) need be notified in respect of the contracting states not indicated under No. 2b.

Prière de ne pas procéder à la signification des notifications prévues par les règles 85bis(1) et 69(1) pour les Etats contractants n'ayant pas été mentionnés au n° 2b.

3. Wird ein **automatischer Abbuchungsauftrag** erteilt (Feld 43), so wird das EPA beauftragt, bei Ablauf der Grundfrist nach Artikel 79(2) den siebenfachen Betrag einer Benennungsgebühr abzubuchen. Ist eine Erklärung unter Nr. 2b abgegeben worden, so sollen die Benennungsgebühren nur für die dort angegebenen Vertragsstaaten abgebucht werden, sofern dem EPA nicht bis zum Ablauf der Grundfrist ein anderslautender Auftrag zugeht.

3. If an **automatic debit order** has been issued (Section 43), the EPO is authorised, on expiry of the basic period under Article 79(2), to debit seven times the amount of the designation fee. If any states are indicated under No. 2b, the EPO shall debit designation fees only for those states, unless it is instructed to do otherwise before expiry of the basic period.

☒ 3. Si un **ordre de prélèvement automatique** est donné (rubrique 43), il est demandé à l'OEB de prélever, à l'expiration du délai normal visé à l'article 79(2), un montant correspondant à sept fois la taxe de désignation. Si une déclaration a été faite au n° 2b, les taxes de désignation ne sont prélevées que pour les Etats contractants qui y sont indiqués, sauf instruction contraire reçue par l'OEB avant l'expiration du délai normal.

* Stand bei Drucklegung: 28 Vertragsstaaten, und zwar: / Status when this form was printed: 28 contracting states, namely / Situation à la date d'impression : 28 Etats contractants, à savoir : **AT** Österreich / Austria / Autriche, **BE** Belgien / Belgium / Belgique, **BG** Bulgarien / Bulgaria / Bulgarie, **CH / LI** Schweiz und Liechtenstein / Switzerland and Liechtenstein / Suisse et Liechtenstein, **CY** Zypern / Cyprus / Chypre, **CZ** Tschechische Republik / Czech Republic / République tchèque, **DE** Deutschland / Germany / Allemagne, **DK** Dänemark / Denmark / Danemark, **EE** Estland / Estonia / Estonie, **ES** Spanien / Spain / Espagne, **FI** Finnland / Finland/Finlande, **FR** Frankreich / France / France, **GB** Vereinigtes Königreich / United Kingdom / Royaume-Uni, **GR** Griechenland / Greece / Grèce, **HU** Ungarn / Hungary / Hongrie, **IE** Irland / Ireland / Irlande, **IT** Italien / Italy / Italie, **LU** Luxemburg / Luxembourg / Luxembourg, **MC** Monaco / Monaco / Monaco, **NL** Niederlande / Netherlands / Pays-Bas, **PL** Polen / Poland / Pologne, **PT** Portugal / Portugal / Portugal, **RO** Rumänien / Romania / Roumanie, **SE** Schweden / Sweden / Suède, **SI** Slowenien / Slovenia / Slovénie, **SK** Slowakische Republik / Slovak Republic / République slovaque, **TR** Türkei / Turkey / Turquie

14738EP /nh

Raum für Zeichen des Anmelders / Space for applicant's reference / Espace réservé à la référence du demandeur

Verschiedene Anmelder für verschiedene Vertragsstaaten /
Different applicants for different contracting states /
Différents demandeurs pour différents Etats contractants

APPR 02 # | | | | | | | | # | | | | | | | | | |

33 Name(n) des (der) Anmelder(s) und benannte Vertragsstaaten /
Name(s) of applicant(s) and designated contracting states /
Nom(s) du (des) demandeur(s) et des Etats contractants désignés

Erstreckung des europäischen Patents

Diese Anmeldung gilt als Antrag, die europäische Patentanmeldung und das darauf erteilte europäische Patent auf alle Nicht-Vertragsstaaten des EPÜ zu erstrecken, mit denen am Tag ihrer Einreichung „Erstreckungsabkommen" bestehen (derzeit: Albanien, Kroatien, Lettland, Litauen, ehemalige jugoslawische Republik Mazedonien). Die Erstreckung wird jedoch nur wirksam, wenn die vorgeschriebene Erstreckungsgebühr entrichtet wird.

Extension of the European patent

This application is deemed to be a request to extend the European patent application and the European patent granted in respect of it to all non-contracting states to the EPC with which "extension agreements" exist on the date on which the application is filed (present situation: Albania, Croatia, Latvia, Lithuania, former Yugoslav Republic of Macedonia). However, the extension only takes effect if the prescribed extension fee is paid.

EXPT

34 **Extension des effets du brevet européen**

☒ La présente demande est réputée constituer une requête en extension des effets de la demande de brevet européen et du brevet européen délivré sur la base de cette demande à tous les Etats non parties à la CBE avec lesquels il existe un «accord d'extension» à la date du dépôt de la demande (situation actuelle : Albanie, Croatie, Lettonie, Lituanie, ex-République yougoslave de Macédoine). Toutefois, l'extension ne produit ses effets que s'il est acquitté la taxe d'extension prescrite.

Es ist derzeit beabsichtigt, die Erstreckungsgebühr für die nachfolgend angekreuzten Staaten zu entrichten: / It is currently intended to pay the extension fee for the states marked below with a cross: / Il est actuellement envisagé de payer la taxe d'extension pour les Etats dont le nom est coché ci-après :

Albanien / Albania / Albanie	AL ☐
Kroatien / Croatia / Croatie	HR ☐
Litauen / Lithuania / Lituanie	LT ☐
Lettland / Latvia / Lettonie	LV ☐
Ehemalige jugoslawische Republik Mazedonien / Former Yugoslav Republic of Macedonia / Ex-République yougoslave de Macédoine	MK ☐
_____	____ ☐
_____	____ ☐
_____	____ ☐

(Platz für Staaten, mit denen nach Drucklegung dieses Formblatts „Erstreckungsabkommen" in Kraft treten) / (Space for states with which "extension agreements" enter into force after this form has been printed) / (Prévu pour des Etats à l'égard desquels des «accords d'extension» entreront en vigueur après l'impression du présent formulaire)

Die Anmeldung ist eine Teilanmeldung / The application is a divisional application / La présente demande constitue une demande divisionnaire

DFIL 9 | | | | | | | | #
PANR | | | | | | | | #

35 ☐ _____ Nummer der früheren Anmeldung / No. of earlier application / Numéro de la demande initiale

Es handelt sich um eine Anmeldung nach Artikel 61(1)b) / The application is an Article 61(1)(b) application / La présente demande constitue une demande selon l'article 61(1)b)

DFIL 9 | | | | | | | | #
EANR | | | | | | | | #

36 ☐ _____ Nummer der früheren Anmeldung / No. of earlier application / Numéro de la demande initiale

Patentansprüche / Claims / Revendications CLMS

37 17 Zahl der Patentansprüche / Number of claims / Nombre de revendications

Zur Veröffentlichung mit der Zusammenfassung wird vorgeschlagen Abbildung Nr. / It is proposed that the abstract be published together with figure No. / Il est proposé de publier avec l'abrégé la figure n° DRAW 2

39 1 Nummer / Number / Numéro

14738EP /nh

Raum für Zeichen des Anmelders / Space for applicant's reference / Espace réservé à la référence du demandeur

Pilotprojekt für europäische Erstanmeldungen (d. h. Anmeldungen, die keine Priorität beanspruchen): Es wird beantragt, daß **kein** erweiterter europäischer Recherchenbericht erstellt wird. / Pilot project for EP first filings (ie applications claiming no priority): It is requested that **no** extended European search report be drawn up. / Projet pilote pour les premiers dépôts (c'est-à-dire des demandes ne revendiquant aucune priorité) : Il est demandé de **ne pas** établir un rapport de recherche européene élargi.

EESR

40 ☐

Zusätzliche Abschrift(en) der im europäischen Recherchenbericht angeführten Schriftstücke wird (werden) beantragt / Additional copy(ies) of the documents cited in the European search report is (are) requested / Prière de fournir une (des) copie(s) supplémentaire(s) des documents cités dans le rapport de recherche européenne

ASOC

41 1

Anzahl der **zusätzlichen** Sätze von Abschriften
Number of **additional** sets of copies
Nombre de jeux **supplémentaires** de copies

Es wird die Rückerstattung der Recherchengebühr gemäß Art. 10 GebO beantragt / Refund of the search fee is requested pursuant to Article 10 of the Rules relating to Fees / Le remboursement de la taxe de recherche est demandé en vertu de l'article 10 du règlement relatif aux taxes

42 ☐

Eine Kopie des Recherchenberichts ist beigefügt / A copy of the search report is attached / Une copie du rapport de recherche est jointe

43 ☐

Automatischer Abbuchungsauftrag
(nur möglich für Inhaber von beim EPA geführten laufenden Konten)

Das EPA wird hiermit beauftragt, fällig werdende Gebühren und Auslagen nach Maßgabe der Vorschriften über das automatische Abbuchungsverfahren vom nebenstehenden laufenden Konto abzubuchen. In bezug auf die **Benennungsgebühren** wird auf Feld 32.3 verwiesen. Das EPA wird ferner beauftragt, die **Erstreckungsgebühren** für jeden in Feld 34 angekreuzten »Erstreckungsstaat« bei Ablauf der Grundfrist zu ihrer Zahlung abzubuchen, sofern ihm nicht bis dahin ein anderslautender Auftrag zugeht.

Automatic debit order
(for EPO deposit account holders only)

The EPO is hereby authorised, under the Arrangements for the automatic debiting procedure, to debit from the deposit account opposite any fees and costs falling due. With regard to **designation fees** reference is made to Section 32.3. The EPO is also authorised, on expiry of the basic period for its payment, to debit the **extension fee** for each of the "extension states" marked with a cross in Section 34, unless it is instructed to do otherwise before expiry of this period.

44

Ordre de prélèvement automatique
(possibilité offerte uniquement aux titulaires de comptes courants ouverts auprès de l'OEB)

Par la présente, il est demandé à l'OEB de prélever du compte courant ci-dessous les taxes et frais venant à échéance, conformément à la réglementation relative à la procédure de prélèvement automatique. Pour les **taxes de désignation,** se reporter à la rubrique 32.3. Il est en outre demandé à l'OEB de prélever, à l'expiration du délai normal prévu pour leur paiement, les **taxes d'extension** pour chaque «Etat autorisant l'extension» coché à la rubrique 34, sauf instruction contraire reçue avant l'expiration de ce délai.

Für automatischen Abbuchungsauftrag:
For automatic debit order:
Pour l'ordre de prélèvement automatique :

DECA

Nummer des laufenden Kontos / Deposit account number / Numéro du compte courant

Name des Kontoinhabers / Account holder's name / Nom du titulaire du compte

Eventuelle **Rückzahlungen** auf das nebenstehende beim EPA geführte laufende Konto / Any **reimbursement** to EPO deposit account opposite / **Remboursements** éventuels à effectuer sur le compte courant ci-contre ouvert auprès de l'OEB

DEPA

Nummer des laufenden Kontos / Deposit account number / Numéro du compte courant

45 2800.0031

Name des Kontoinhabers / Account holder's name / Nom du titulaire du compte

Kraus & Weisert

Die vorgeschriebene Liste über die diesem Antrag beigefügten Unterlagen ergibt sich aus der vorbereiteten Empfangsbescheinigung (Seite 6 dieses Antrages)

The prescribed list of documents enclosed with this request is shown on the prepared receipt (page 6 of this request)

46

La liste prescrite des documents joints à cette requête figure sur le récépissé préétabli (page 6 de la présente requête)

Unterschrift(en) des (der) Anmelder(s) oder Vertreter(s) / Signature(s) of applicant(s) or representative(s) / Signature(s) du (des) demandeur(s) ou du (des) mandataire(s)

47

Für Angestellte nach Artikel 133(3) Satz 1 mit allgemeiner Vollmacht / For employees under Article 133(3), first sentence, having a general authorisation / Pour les employés mentionnés à l'article 133(3), 1ère phrase, munis d'un pouvoir général

Nr. / No. / n° :

Ort / Place / Lieu Munich

Datum / Date September 15, 2004

Dipl.-Ing. Hans-Jörg Banzer

Kraus & Weisert
Patent- und Rechtsanwälte
Dr. T. Albrecht · Dipl.-Ing. H.-J. Banzer · Dr. H. Adam
Dr. I. Hiebl · Dr. C. Beckmann · Dr. F. Nielsen
Thomas-Wimmer-Ring 15 • 80539 München

Name des (der) Unterzeichneten bitte in Druckschrift wiederholen. Bei juristischen Personen bitte die Stellung des (der) Unterzeichneten innerhalb der Gesellschaft in Druckschrift angeben. / Please print name under signature. In the case of legal persons, the position of the signatory within the company should also be printed. / Le ou les noms des signataires doivent être indiqués en caractères d'imprimerie. S'il s'agit d'une personne morale, la position occupée au sein de celle-ci par le ou les signataires doit être indiquée en caractères d'imprimerie.

14738EP /nh

Raum für Zeichen des Anmelders / Space for applicant's reference / Espace réservé à la référence du demandeur

14738EP /nh

Empfangsbescheinigung / Receipt for documents / Récépissé de documents

6

(Liste der diesem Antrag beigefügten Unterlagen) (Checklist of enclosed documents) (Liste des documents annexés à la présente requête)

Es wird hiermit der Empfang der unten bezeichneten Dokumente bescheinigt / Receipt of the documents indicated below is hereby acknowledged / Nous attestons le dépôt des documents désignés ci-dessous

Wird im Falle der Einreichung der europäischen Patentanmeldung bei einer nationalen Behörde diese Empfangsbescheinigung vom Europäischen Patentamt übersandt, so ist sie als Mitteilung gemäß Regel 24(4) anzusehen (siehe Feld RENA). **Nach Erhalt der Mitteilung nach Regel 24(4) sind alle weiteren Unterlagen, die die Anmeldung betreffen, nur noch unmittelbar beim EPA einzureichen.** / If this receipt is issued by the European Patent Office and the European patent application was filed with a national authority it serves as a communication under Rule 24(4) (see Section RENA). **Once the communication under Rule 24(4) has been received, all further documents relating to the application must be sent directly to the European Patent Office.** / Si, en cas de dépôt de la demande de brevet européen auprès d'un service national, l'Office européen des brevets délivre le présent récépissé de documents, ce récépissé est réputé être la notification visée à la règle 24(4) (cf. rubrique RENA). **Dès que la notification visée à la règle 24(4) a été reçue, tous les autres documents relatifs à la demande doivent être adressés directement à l'OEB.**

KRAUS & Weisert
Thomas-Wimmer-Ring 15
80539 Munich

Nur für amtlichen Gebrauch / For official use only / Cadre réservé à l'administration

Datum / Date

Europäisches Patentamt
European Patent Office
Office européen des brevets
D-80298 München
M. Daniel S. Camara

Unterschrift / Amtsstempel / Signature / Official stamp / Signature / Cachet officiel

Anmeldenummer / Application No. / N° de la demande			04021916.4
Tag des Eingangs (Regel 24(2)) / Date of receipt (Rule 24(2)) / Date de réception (règle 24(2))	DREC		15.09.04
Zeichen des Anmelders/Vertreters / Applicant's/ Representative's ref. / Référence du demandeur ou du mandataire	AREF		14738EP /nh
Nur nach Einreichung der Anmeldung bei einer nationalen Behörde: / Only after filing of the application with a national authority: / Seulement après le dépôt de la demande auprès d'un service national:			
Tag des Eingangs beim EPA (Regel 24(4)) / Date of receipt at EPO (Rule 24(4)) / Date de réception à l'OEB (règle 24(4))	RENA		

			Blattzahl* / Number of sheets* / Nombre de feuilles*	Gesamtzahl der Abbildungen* / Total number of figures* / Nombre total de figures*
A. Anmeldungsunterlagen und Prioritätsbeleg(e) / Application documents and priority document(s) / Pièces de la demande et document(s) de priorité		48		
1. Beschreibung (ohne Sequenzprotokollteil) / Description (excluding sequence listing part) / Description (sauf partie réservée au listage des séquences)			X 30	
2. Patentansprüche / Claim(s) / Revendication(s)			X 5	
3. Zeichnung(en) / Drawing(s) / Dessin(s)	DRAW 1 #		X 17	17
4. Sequenzprotokollteil der Beschreibung / Sequence listing part of description / Partie de la description réservée au listage des séquences				
5. Zusammenfassung / Abstract / Abrégé			X 1	
6. Übersetzung der Anmeldungsunterlagen / Translation of the application documents / Traduction des pièces de la demande				
7. Prioritätsbeleg(e) / Priority document(s) / Document(s) de priorité				
8. Übersetzung des (der) Prioritätsbelegs(belege) / Translation of priority document(s) / Traduction du (des) document(s) de priorité				

* Die Richtigkeit der Angabe der Blattzahl und der Gesamtzahl der Abbildungen wurde bei Eingang nicht geprüft / No check was made on receipt that the number of sheets and the total number of figures indicated were correct / L'exactitude du nombre de feuilles et du nombre total de figures n'a pas été contrôlée lors du dépôt

B. Der Anmeldung in der eingereichten Fassung liegen folgende Unterlagen bei: / This application as filed is accompanied by the items below: / A la présente demande sont annexées les pièces suivantes:		49	
1. Einzelvollmacht / Specific authorisation / Pouvoir particulier			
2. Allgemeine Vollmacht / General authorisation / Pouvoir général			
3. Erfindernennung / Designation of inventor / Désignation de l'inventeur			
4. Früherer Recherchenbericht / Earlier search report / Rapport de recherche antérieure			
5. Gebührenzahlungsvordruck (EPA Form 1010) / Voucher for the settlement of fees (EPO Form 1010) / Bordereau de règlement de taxes (OEB Form 1010)			X
6. Scheck (nicht bei Einreichung bei den nationalen Behörden) / Cheque (not when filing with national authorities) / Chèque (pas de chèque en cas de dépôt auprès des services nationaux)			X
7. Datenträger für Sequenzprotokoll / Data carrier for sequence listing / Support de données pour liste de séquences	SEQL 4		
8. Zusatzblatt / Additional sheet / Feuille supplémentaire			
9. Sonstige Unterlagen (bitte hier spezifizieren) / Other documents (please specify here) / Autres documents (veuillez préciser)			

Währung Betrag / Currency Amount / Monnaie Montant
(Ausfüllung freigestellt / optional / facultatif)
EUR 1115,00

C. Kopien dieser Empfangsbescheinigung / Copies of this receipt for documents / Copies du présent récépissé de documents		50	2 Anzahl der Kopien / Number of copies / Nombre de copies